ELEVEN VIEWS

FREE RESOURCE

The Token Diet

How to get Claude to build 30 to 40 Remotion scenes in a single session

Most people run out of Claude before they run out of video. You start strong, Claude builds scene 1, scene 2, scene 3, the motion graphics look great, and then somewhere around scene 9 it slows down, starts re-explaining what it already built, asks you to confirm things you settled twenty minutes ago, and then the session limit hits and you are starting over with a model that has forgotten everything. That is not Claude being weak. That is you feeding it like a junior dev who needs every detail respelled, instead of feeding it like a production line that already knows the parts. We build 30 to 40 Remotion scenes per session at the Desk on a normal usage plan. The difference is not a secret model or a bigger plan. It is what you put into the context window and what you keep out. This is the whole method, with the prompts.

atlas.elevenviews.io

01

Why Claude quits at scene 9

Claude does not have a scene counter. It has a context window, and every token you spend on one thing is a token you cannot spend on the next scene. When the window fills, two things happen. First, the model gets slower and vaguer because it is re-reading a giant transcript before every reply. Second, the session hits its usage ceiling and ends.

The naive build burns the window in four predictable places. Knowing where the leaks are is half the fix.

1. Re-explained context. Every time you paste your brand colors, fonts, and animation style again, you pay for it again. By scene 9 you have paid for your color palette nine times.

2. Full code echoes. Claude writes a 120-line scene component, then in the next message it quotes large chunks of that file back to you to explain what it did. That echo is pure waste. You already have the file.

3. Re-derived primitives. If every scene reinvents its own title animation, its own background, its own lower-third, Claude rewrites the same 40 lines of spring and interpolate logic over and over. Forty lines times forty scenes is your whole window gone on duplication.

4. Open-ended back-and-forth. Long debates about whether the kicker should fade or slide cost hundreds of tokens per round and produce nothing renderable.

The fix for all four is the same idea: decide once, build reusable parts once, and make every later message tiny. That is the diet.

02

Lock the creative direction before scene 1

The single biggest leak is re-explaining your look. So stop re-explaining it. Spend your first message establishing a creative-direction lock that Claude treats as fixed law for the rest of the session, then never restate it.

The lock is a compact spec: colors as hex, two fonts, timing defaults in frames, motion language in plain words, and the render target. Compact is the point. You are not writing a brand book. You are writing the smallest set of rules that removes ambiguity so Claude never has to ask and you never have to repeat.

The trick that saves the most tokens: tell Claude to acknowledge the lock in one line and never echo it back in full again. A junior dev repeats the brief to prove they understood. You do not want that. You want a worker who absorbed the brief and now just builds.

Once the lock is set, your per-scene prompts can be one or two sentences because the look is already decided. "Scene 12, three stat cards counting up" is enough, because Claude already knows the card

style, the count animation, the colors, and the duration. That is the difference between a 400-token scene request and a 40-token one.

03

Build the primitives first, then reference them by name

Before you ask for a single finished scene, have Claude build a small library of shared components. This is the highest-leverage move in the whole method, because it turns every future scene from "write 120 lines" into "compose 4 named parts."

A Remotion video has maybe 8 to 12 building blocks that show up again and again. Build them once as named, reusable components:

- A background component (gradient, grain, or solid) that every scene sits on.
- A title primitive with the standard entrance animation baked in.
- A subtitle or kicker primitive.
- A lower-third for names and labels.
- A stat or number card with a count-up.
- A bullet list that staggers in.
- An image frame with a consistent mask, border, and entrance.
- A logo sting for intro and outro.
- A transition wrapper for scene-to-scene continuity.

Get these into a single primitives file. Now every scene references them by name. Instead of Claude re-deriving spring physics for the fortieth title, it writes `<Title>Q3 revenue</Title>` and moves on. The token cost of a scene collapses, the visual consistency goes up automatically because every scene pulls from the same parts, and your edits become global: change the title primitive once and all 40 scenes update.

This is also why your renders look like one video instead of forty different ones. Shared parts are shared style.

04

The build, confirm, next loop

The thing that quietly destroys long sessions is conversation. Every time Claude pauses to explain, justify, or re-quote, you lose window and momentum. So run a tight loop that keeps it shipping.

The loop is three beats. Build: Claude writes the scene component, full file, nothing else. Confirm: you reply with one short signal, usually a single word or a one-line tweak. Next: Claude moves to the next scene without recapping the last one.

The rules that make the loop hold:

- No echoing. Claude writes the file and a one-line note, not a paragraph explaining the file.
- No unprompted recaps. It does not summarize scenes 1 through 11 before building 12.

- Tweaks are surgical. "Title 6 frames slower" not "can you adjust the timing a bit."
- You drive the index. You say the scene number and the content. Claude does not ask what is next; it waits for you.

Keep a running scene list in your own notes, not in the chat. The chat is for building. Your notes are the source of truth for what is built and what is left. This keeps the window clean and means you can resume instantly if the session ends.

When this loop is running well it feels like a press, not a conversation. You feed a line, a scene comes out, you feed the next line. That cadence is what gets you to scene 38.

05

Resume at scene 21 with full continuity

You will sometimes hit a limit mid-build. That is fine if you can resume without rebuilding. The reason most people start over is that the new session has no memory of the lock, the primitives, or where they left off. Give it all three in one compact resume prompt and the new session picks up cold at scene 21 like nothing happened.

A good resume prompt carries four things and nothing more: the creative-direction lock, the list of primitives that already exist (by name and signature, not full code), the scenes already built, and the next scene to build. You are handing the new session a memo, not a transcript.

The key discipline: do not paste the old chat. Pasting the transcript reintroduces every leak you avoided the first time. Paste the distilled state. A 300-token memo restores continuity that a 30,000-token transcript would, at a hundredth of the cost, and it leaves the new window almost entirely free for actual building.

Keep the primitives file in your project, not the chat. The new session reads it from disk. The resume prompt just needs to know it exists and what is in it.

06

Reading the drift signals

Claude tells you when it is about to fall apart, if you watch for it. Catch the signals early and you decide consciously whether to push the current session or start fresh, instead of being surprised by a limit.

The signals that mean drift is starting:

- It re-explains a primitive you already locked. The window is getting crowded and it is re-deriving instead of referencing.
- It asks you to confirm something already decided. The lock is slipping out of effective context.
- Replies get longer and vaguer. More hedging, more recap, less code.
- Small visual inconsistencies creep in: a slightly different easing, a color that drifted off the lock. It is reconstructing from memory instead of from the primitives.
- It starts apologizing or summarizing unprompted. That is the model trying to re-anchor itself.

When you see one signal, finish the current scene and tighten your prompts. When you see two or three, do not fight it. Stop, save your scene list, and fire the resume prompt into a fresh session. A clean window at scene 22 ships faster than a clogged one at scene 22. Pushing a drifting session to squeeze out two more scenes usually costs you more time in fixing inconsistencies than a fresh start would have.

07

The numbers: naive vs the diet

Same model, same plan, same video brief. The only variable is method.

Naive prompting. You paste the full brand spec on every scene. Claude writes each scene from scratch with its own inline animation logic. After each file it explains what it did in a paragraph and quotes the code back. You debate small choices in the chat. Result: the window fills fast, replies slow down around scene 6, and the session ends around scene 8 or 9. Call it 8 finished scenes, and several of them drift off-style because the spec degraded.

The diet. One creative-direction lock up front, acknowledged in a line. A primitives file built before scene 1. Then the build, confirm, next loop with no echoes and surgical tweaks, your scene list kept in your own notes. Result: each scene request is tiny, each reply is a file plus a one-liner, and the window stays lean. You get to roughly 38 scenes in the same session before you would even consider a resume, and they all look like one video because they share parts.

Eight versus thirty-eight. That is not a small optimization. On a usage plan that is the difference between a one-coffee afternoon and a five-session slog spread across three days, for the exact same finished video. The model was never the bottleneck. The feed was.

08

Build Kit: turn this guide into video

This guide is not just an explainer. It is the build spec for its own companion video. Everything the Desk needs to produce the Part 1 video is already in this PDF: the four context eaters, the shared-primitives pattern, the build-confirm-next loop, the drift signals, and the 8-vs-38 comparison. The Build Kit below tells you exactly how the source becomes the screen.

The workflow mirrors the very thing this guide teaches. You do not ask Claude to build the whole video in one shot. You feed it the shared primitives first, then build scenes one at a time, confirming each before moving to the next. The companion video is itself a Remotion project produced on a token diet, so the production is a working demo of the method.

Hand the Desk three things and it can build: (1) this PDF as the source of truth, (2) the narration script from the swipeables below, and (3) the scene map. The Desk reads the scene map, sets up the shared primitives once (the Scene wrapper, the color tokens, the type ramp, the timing constants), then walks the build-confirm-next loop down the scene list. When the session nears its context limit, it uses the resume prompt pattern from the guide: state which scene was last confirmed, restate the shared primitives by name, and continue from the next row.

The order of operations:
1. Lift the four diet rules and the shared-primitives list straight from the guide. These become the Remotion primitives file.
2. Paste the scene map into the build session. It is already in build-confirm-next order.
3. Build the primitives first, confirm they render, then build scenes one row at a time.
4. Watch for the drift signals named in the guide. If Claude starts re-declaring colors per scene or re-explaining the timing helper, that is the cue to compact context or resume.
5. Record the voiceover from the Part 1 script, drop it on the audio track, and align scene boundaries to the narration beats marked in the scene map.
6. Render at the settings in the Render and export recipe.

The point: one PDF, two outputs. The free guide teaches the method. The same spec, read by the Desk, ships the paid video that demonstrates it.

09

Your companion video guides

The paid track is a three-part video guide that takes a reader from "Claude quits at scene 8" to a finished 38-scene Remotion render produced in a single session. Each part is built by the Desk from this same PDF, so what you watch on screen is the method running live, not slides about the method.

Part 1: Why Claude quits at scene 8 (4 to 6 minutes). On screen you see a real session hitting the wall: context filling, scenes getting sloppier, then the hard stop. The video names the four things eating the window using the guide's own list, and shows the shared-primitives pattern as a single primitives file that every scene imports. You walk away understanding why the limit hits and having the primitives file structure to copy.

Part 2: The build-confirm-next loop, live (8 to 12 minutes). This is a screen recording of an actual build session. You watch the operator set up primitives once, then build scenes one at a time, confirming each render before the next prompt. You see the exact phrasing that keeps Claude lean and the drift signals appearing in real time, plus the correction that pulls it back. You walk away able to run the loop yourself, having seen ten-plus scenes built without a single context reset.

Part 3: Hitting the limit and resuming clean (6 to 9 minutes). The session is deliberately pushed to its limit. You watch the resume prompt do its job: restate the confirmed scene, name the shared primitives, continue from the next row. The video ends on the 8-vs-38 comparison rendered as a real Remotion sequence, side by side. You walk away with the resume prompt as a reusable template and a complete 38-scene project file you can open and study.

Honest scope: the track teaches token discipline for long Remotion builds with Claude. It does not teach Remotion from zero, motion design theory, or audio production. It assumes you can run a Remotion project and write a basic component. What it gives you that you cannot get from the free PDF: the live sessions, the exact prompt phrasing as spoken, and the finished 38-scene project to keep.

Swipe file

Copy, paste, adjust. These are the exact prompts and templates.

CREATIVE-DIRECTION LOCK (PASTE FIRST, ONCE)

You are building Remotion scene components for one video. Lock this creative direction as fixed for the entire session. Acknowledge it in ONE line, then never restate it in full again.

Resolution: 1920x1080, 30fps. (Switch to 3840x2160 for 4K finals.)
Colors: background #0A0A0B, primary text #F4F1EA, accent #C9A24B, muted #8A8A8A.
Fonts: display = Fraunces (titles), body = Inter (everything else).
Motion language: things ease in with a spring, settle, hold. No bounce, no spin.
Entrances ~12 frames, holds vary by scene.
Defaults: title enters frames 0–12, content staggers after, scene length I will specify per scene.
Style: clean, editorial, confident. Generous spacing. One idea per scene.

From now on I will request scenes by number and content only. Do not echo this spec back, do not re-explain it. Confirm once and wait for scene 1.

PRIMITIVES BUILD REQUEST (BEFORE ANY SCENE)

Before we build any scenes, create a single primitives file (Primitives.tsx) with reusable components I can compose. Use the locked creative direction for all defaults. Build:

- <Background variant="gradient" | "solid" | "grain">
- <Title> with the standard 12-frame spring entrance
- <Kicker> small label above titles
- <LowerThird name= role= >
- <StatCard value= label= > with a count-up
- <BulletList items={[]} > staggered entrance
- <ImageFrame src= > consistent mask, border, entrance
- <LogoSting> for intro/outro
- <SceneTransition> wrapper for continuity

Write the full file once. After that, every scene should import and compose these by name. Do not re-derive animation logic inside scenes. One-line note when done, then wait for scene 1.

PER-SCENE REQUEST (TINY, REPEATABLE)

Scene 12, 90 frames. Kicker "Q3 RESULTS", title "Revenue up 38%", then three StatCards counting up: 38% growth, $2.4M ARR, 11k users. Standard transition out. Build the file, one-line note, wait.

SURGICAL TWEAK FORMAT

Scene 12: StatCards enter 6 frames later and stagger by 4 frames. Title hold +10 frames. Everything else stays. Reship the file only.

RESUME PROMPT (AFTER HITTING A LIMIT)

```
We are resuming a Remotion build. Here is the full state. Do not ask me to re-explain anything.

CREATIVE LOCK: 1920x1080 30fps. bg #0A0A0B, text #F4F1EA, accent #C9A24B, muted #8A8A8A. Display Fraunces, body Inter. Spring entrances ~12 frames, no bounce/spin, clean editorial.

PRIMITIVES (already exist in Primitives.tsx, read from disk, do not rewrite): Background, Title, Kicker, LowerThird, StatCard, BulletList, ImageFrame, LogoSting, SceneTransition.

BUILT: scenes 1–20, all shipped and approved.
NEXT: scene 21, 120 frames, title "How the Desk works", BulletList of 4 steps, ImageFrame on the right.

Confirm in one line and build scene 21. Same loop as before: file plus one-line note, no echoes, I drive the scene index.
```

ONE-PAGE SESSION-SETUP TEMPLATE

```
VIDEO: [name]   TARGET: [1080p / 4K]   FPS: 30

CREATIVE LOCK
- bg / text / accent / muted hex:
- display font / body font:
- motion language (one line):
- entrance default (frames):

PRIMITIVES TO BUILD FIRST
[ ] Background  [ ] Title  [ ] Kicker  [ ] LowerThird
[ ] StatCard  [ ] BulletList  [ ] ImageFrame  [ ] LogoSting  [ ] SceneTransition

SCENE LIST (keep in YOUR notes, not the chat)
01 ____  02 ____  03 ____  ...  (number + one-line content each)

LOOP RULES
- file + one-line note only, no echoes
- I drive the scene index, Claude waits
- tweaks are surgical (scene #, exact change)

DRIFT WATCH (resume fresh when 2+ appear)
[ ] re-explains a locked primitive
[ ] asks to confirm a settled decision
[ ] replies getting longer/vaguer
[ ] style inconsistencies creeping in

NEXT STOP: DaVinci Resolve import / green-screen removal / 4K render
```

COMPANION VIDEO NARRATION SCRIPT (PART 1)

[Cold open, screen recording of a Claude session mid-build]

Watch this. Claude is building Remotion scenes. Scene one, clean. Scene four, still good. Scene eight, look at the code now. The colors are getting re-declared. The timing math is drifting. And then it stops. Out of room. You wanted thirty-eight scenes. You got eight.

This is not a Claude problem. This is a feeding problem. You put the whole project on its plate at once, and the context window filled before the work was done.

So let us put Claude on a diet.

[Cut to title card: The Token Diet]

Four things eat your context window on a long Remotion build. One: repeated boilerplate, the same imports and wrappers pasted into every scene. Two: re-declared styling, the same colors and fonts spelled out scene after scene. Three: re-explained logic, Claude reminding itself how the timing helper works every time. Four: accumulated chat, the full back and forth of every prior scene sitting in the window.

Each one is avoidable. And the fix for all four is the same move.

[Cut to a single file on screen: primitives.tsx]

Shared primitives. You define the things every scene needs exactly once. One Scene wrapper. One set of color tokens. One type ramp. One timing constant. Every scene imports them. Claude never re-declares them, because they already exist.

This is the difference between feeding Claude the whole meal and feeding it one bite at a time off a plate that is already set.

[Cut to scene list scrolling]

The primitives go in first. Then you build one scene. You confirm it renders. Then the next. Build, confirm, next. The window stays lean because each prompt only carries one scene of new work, not the weight of all the scenes before it.

[Cut to the 8 vs 38 comparison, side by side]

Here is the same project, two ways. On the left, everything in one prompt: eight scenes, then the wall. On the right, shared primitives plus the build-confirm-next loop: thirty-eight scenes, one session, no reset.

Same model. Same machine. Different diet.

[Close]

In the next part, you watch the loop run live, start to finish, and you see the exact words that keep Claude lean. Grab the full guide. Then come build.

SCENE MAP FOR THE COMPANION VIDEO

scene | on-screen text | motion note | asset needed
1 | (none, raw session footage) | screen recording plays at 1x, slight zoom-in on the cursor | clip: claude-session-stall.mp4
2 | Scene 8. Out of room. | hard cut, text snaps in with a 2-frame shake | primitives: Scene wrapper, type ramp
3 | This is a feeding problem | fade up, word "feeding" weighted in gold | color token: gold #C9A24B
4 | The Token Diet | title scales from 0.9 to 1.0, hold | logo lockup, Fraunces display font
5 | 4 things eat your window | four list rows stagger in, 4 frames apart | primitives: ListRow component
6 | 1 Repeated boilerplate | row highlights, faint code ghost behind | bg asset: blurred-code.png
7 | 2 Re-declared styling | same ListRow, swap label only | (reuse scene 6 primitives)
8 | 3 Re-explained logic | same ListRow, swap label only | (reuse scene 6 primitives)
9 | 4 Accumulated chat | same ListRow, swap label only | (reuse scene 6 primitives)
10 | One fix for all four | rows collapse into a single centered line | primitives: timing constant FADE=12
11 | primitives.tsx | file name types on, mono font | mono font: JetBrains Mono
12 | Define once. Import everywhere. | two-line stack, second line slides up | (reuse type ramp)
13 | Build. Confirm. Next. | three words pulse in sequence, 8 frames apart | primitives: Pulse helper
14 | One scene per prompt | scene list scrolls, one row lights at a time | clip: scene-list-scroll.mp4
15 | 8 scenes | left panel fills to 8, then stops with a red cap | primitives: BarMeter component
16 | 38 scenes | right panel fills past 8 to 38, green, no cap | (reuse BarMeter primitives)
17 | Same model. Same machine. | two lines, low weight, centered | (reuse type ramp)
18 | Different diet. | gold weight on "diet", slow fade | color token: gold #C9A24B
19 | Watch the loop run live. Next part. | CTA card slides up from bottom | logo lockup, gold CTA chip
20 | atlas.elevenviews.io | url fades in under CTA, hold 90 frames | logo lockup

ASSET AND RESOURCE CHECKLIST

SOURCE OF TRUTH
- The Token Diet PDF (this document) — the build spec the Desk reads
- Part 1 narration script (swipeable above) — voiceover source
- Scene map (swipeable above) — build order, already in build-confirm-next sequence

REMOTION PROJECT FILES
- primitives.tsx — Scene wrapper, color tokens, type ramp, timing constants. Built first, before any scene. This IS the shared-primitives pattern from the guide.
- Components defined in primitives and reused across scenes: ListRow, BarMeter, Pulse helper
- Composition file registering the 20-scene sequence at 30fps

FONTS
- Fraunces (display, headlines and title cards) — variable weight
- Inter (body and small labels)
- JetBrains Mono (file names, code ghosts) — scenes 11 and 6

COLOR TOKENS (declared once in primitives)
- Near-black background #0A0A0B
- Gold accent #C9A24B
- Off-white text #F4F2EC
- Alert red (8-scene cap) and confirm green (38-scene bar)

VIDEO AND IMAGE ASSETS
- claude-session-stall.mp4 — real screen recording of a session hitting the limit (scene 1)
- scene-list-scroll.mp4 — screen recording of the scene list (scene 14)
- blurred-code.png — background ghost behind list rows (scene 6)
- Eleven Views logo lockup, transparent PNG (scenes 4, 19, 20)

AUDIO
- Recorded voiceover from the Part 1 script, single take per beat, WAV 48kHz
- Optional low ambient bed under VO, ducked 18dB below voice

PROMPT DOCS FOR THE DESK
- The build-confirm-next loop phrasing (lift from the guide)
- The resume prompt template (lift from the guide) — used if the build session nears its own limit
- The drift-signal list (lift from the guide) — the Desk watches for these while building

RENDER AND EXPORT RECIPE

ORDER OF OPERATIONS
1. Build primitives.tsx first. Render a single test scene to confirm tokens, fonts, and timing constants resolve. Do not build scene 1 until primitives render clean.
2. Build scenes 1 through 20 with the build-confirm-next loop, one scene per prompt. Confirm each renders before the next.
3. Drop the voiceover WAV on the audio track. Align scene boundaries to the narration beats: scene 4 title lands on "The Token Diet," scenes 15 and 16 land on "eight scenes" and "thirty-eight scenes."
4. Add the optional ambient bed, ducked under voice.
5. Render a 1080p draft first to check timing and alignment. Only render 4K once the draft is approved.

1080p DRAFT AND SOCIAL CUT
- Resolution 1920x1080, 30fps
- Codec H.264, CRF 18, preset medium
- Pixel format yuv420p (universal playback)
- Audio AAC 192kbps 48kHz
- Remotion: npx remotion render TokenDiet out/token-diet-1080.mp4 --codec=h264 --crf=18

4K MASTER
- Resolution 3840x2160, 30fps
- Codec H.265 (HEVC), CRF 20, OR ProRes 422 HQ for a true master

```
- Pixel format yuv420p10le if HEVC, for 10-bit headroom
- Audio AAC 256kbps 48kHz
- Scale the composition with --scale=2 from the 1080 comp rather than rebuilding layouts
- Remotion HEVC: npx remotion render TokenDiet out/token-diet-4k.mp4 --codec=h265 --scale=2 --crf=20
- Remotion ProRes master: npx remotion render TokenDiet out/token-diet-4k.mov --codec=prores --prores-profile=4444 --scale=2

FINISH
- Verify audio sync on the 4K master at scenes 4, 15, 16 (the beat-aligned moments)
- Export a 1080 vertical 9:16 crop for social if needed: re-render with a 1080x1920 composition reusing the same primitives, do not stretch the 16:9 master
- Keep the ProRes .mov as the archive master, ship the H.264 1080 and H.265 4K as delivery files
```

Want this built into your business?

If you want the finished videos without running the press yourself, that is what Eleven Views does. The Desk is about 25 Claude-powered agents that turn scripts into motion-graphic scenes, full landing pages, PDFs, and sales funnels at roughly a twentieth of typical cost. Book a build at elevenviews.io and we will produce your video or funnel start to finish, or license the Desk and run this method at volume in-house. Either way, you stop quitting at scene 9. Book a call at atlas.elevenviews.io/book.