25 Video Effect Templates: Build Your Own Library With Claude Code

A walkthrough for turning any reel effect you spot into reusable code you can drop into your next video, the way I built mine.

Why I built this

I kept seeing the same handful of effects everywhere: a camera that pushes in while the background stretches, a spotlight that reveals part of a frame, giant captions that fill the screen. Every time I wanted one of those looks, I rebuilt it from scratch in whatever editor I had open that day.

So I started asking Claude Code to build each effect once, as a small piece of reusable code, instead of as a one-off edit. Twenty five effects later, I had a repo full of templates I could call on any clip in seconds.

This guide is that process. Not the finished repo, the method, so you can build your own library around the effects you actually use, in whatever style fits your content. If you want to see the finished result first, my 25 effects are free and open source at github.com/nessalazne/claude-reel-effects.

The hardest part was never getting Claude Code to produce a nice-looking effect on the first try. It usually does. The hard part was getting into the habit of asking for a template instead of an edit, every single time, even when I only needed the effect once. That one-word difference in how you phrase the request is most of this guide.

01

How it actually works

1. Watch the reel or TikTok that has the effect, and describe in plain language what moves and what stays still, as if you were explaining it to someone who cannot see the screen. A vertigo effect is the camera pushing forward while the background stretches outward. A lockdown effect is the opposite: the frame holds still while something small moves inside it. Getting this description right matters more than anything else in the process, since Claude Code is only as accurate as the motion you describe to it.
2. Ask Claude Code to build it as a template, not a one-off edit. The wording matters: a request for 'a template that takes any clip and renders the vertigo zoom' produces a reusable file, while 'zoom in on this video' produces a single edit you cannot reuse. Say out loud what the inputs should be (a clip, a duration, maybe a piece of text) so the first version is already shaped like something you can call again.
3. Save the result to a templates folder and run it on a second, unrelated clip before you trust it. A template only counts once it works on footage it was never written for. If the second run looks wrong, that is useful information: it tells you exactly which assumption from the first clip got baked in by accident.

02

The one thing almost everyone gets wrong

Treating the first working version as done. An effect that only works on the exact clip you tested it on is still a one-off, it just looks like a template because the code is sitting in a file.

> **WATCH OUT**
>
> Before you move on to the next effect, run the one you just built on a clip with a different aspect ratio, length, and subject. If it breaks, that is Claude Code's next fix, not a reason to start over.

---

03 The effects worth building first

EFFECT	WHAT IT LOOKS LIKE
Vertigo	Camera pushes in while the background stretches out, the classic dolly zoom
Orbit	The camera appears to circle around a subject or object
Lockdown	The frame stays still while motion happens inside it
Spotlight	A circle of light reveals part of the frame, the rest stays dark
Backdrop Blur	The background softens so the subject reads sharper
Digital Zoom	A fast punch-in zoom on a detail, often with a frame flash
Rabbit Hole	A zoom-through transition that pulls the viewer into the next scene
Paper Cut	A cutout, collage-style wipe between shots
Glass Widgets	UI-style glass panels and widgets overlaid on the footage, like a phone interface floating on screen
Shadow Puppet	A silhouette or cutout shape moves across the frame like a shadow puppet
Reject Stamp	A stamped graphic slams onto the frame, timed to a beat or a word
Giant Captions	Oversized, screen-filling text synced to a word or phrase
Countdown	An on-screen countdown timer overlay

Grab my free 25 effects

Want to start from a working library instead of an empty folder? I put all 25 effects from this guide in a public repo. It is free to use, so you can run them on your own clips and study how each one is built before making your own.

> **FREE REPO**
>
> - github.com/nessalazne/claude-reel-effects
> **https://github.com/nessalazne/claude-reel-effects**

Set it up in Claude Code or Codex

1. Create a folder for the library, for example ~/video-templates/, with one subfolder per effect so each template stays self-contained and nothing from one effect leaks into another's code.
2. If you are using Claude Code, keep a short README.md at the root of the folder listing each effect's name and what inputs it expects (clip path, text, duration, color). That description is what lets you ask for 'the spotlight template on this new clip' weeks later and get the same result, instead of re-explaining the effect from scratch.
3. If you are using Codex instead, the same folder works as a project directory, or point to it from AGENTS.md if it lives alongside other code, so Codex knows the templates exist before you ask it to use one.
4. Pick the rendering tool that matches the effect. Motion-heavy effects (vertigo, orbit, rabbit hole) are easiest to build as Remotion components, since Remotion handles camera-style movement and timing as React code you can preview frame by frame. Simpler overlays (giant captions, countdown, reject stamp) can be a direct ffmpeg filter chain with no extra framework, which renders faster and has fewer moving parts to maintain.
5. Build and test one effect fully, end to end on two different clips, before starting the next one. A library of ten reliable templates beats twenty five that only work once, and it is much easier to debug one template at a time than to untangle five half-finished ones later.
6. Once a handful of templates are solid, start keeping a short changelog of what changed each time you fix one. Effects tend to get revisited months later when a platform changes its aspect ratio or export settings, and a changelog saves you from re-diagnosing the same issue twice.

> 💡 **QUICK TIPS**
>
> - Name each template file after the effect in plain English (vertigo.tsx, spotlight.tsx) so you can ask for it by name months later without opening the file first.
> - Keep one sample clip per effect in the folder too, so you always have a known-good test case to check a template against after any change.

04

The prompts I actually use

NAMING A NEW EFFECT FROM A REEL

```
I watched a reel where the camera seems to push forward on the subject while the background stretches outward and gets distorted at the edges. Build this as a reusable video effect template, not a one-off edit on a specific clip. Call it vertigo. It should accept any input clip and a duration, and apply the same push-stretch effect regardless of what is in the frame.
```

TURNING A ONE-OFF EDIT INTO A TEMPLATE

```
You just built [EFFECT NAME] for [SPECIFIC CLIP]. Refactor it into a reusable template that takes a clip path, a duration, and any text as parameters, instead of hardcoding this specific clip's values. Save it as its own file in the templates folder.
```

TESTING A TEMPLATE ON NEW FOOTAGE

```
Run the [EFFECT NAME] template on [NEW CLIP PATH], a clip with a different aspect ratio and length than the one it was built on. If it breaks or looks wrong, fix the template itself, not just this one output.
```

DOCUMENTING A FINISHED TEMPLATE

```
Add an entry to README.md for the [EFFECT NAME] template: what it looks like on screen, what inputs it expects, and one example command or call that renders it.
```

PULLING A REUSABLE PIECE OUT OF TWO DIFFERENT EFFECTS

```
I have two templates, [EFFECT A] and [EFFECT B], that both do a version of a zoom but with different easing. Pull the shared zoom logic into one helper function both templates can call, instead of keeping two separate copies of nearly the same code.
```

When it breaks

ERROR	FIX
Effect looks right on the original clip but breaks on a new one	The template still has hardcoded values from the first clip (resolution, duration, color). Ask Claude Code to replace them with parameters.
Claude Code rebuilds the effect from scratch instead of reusing the template file	Point it directly at the template's file path in your request instead of describing the effect again from memory.
Remotion render fails after adding a new template	Check that the new component is registered in the root composition file; a template that exists as a file but is not registered will not render.
Effect works in preview but looks different in the final export	Confirm the export resolution and frame rate match what the template was built and tested against.
Two templates end up with nearly identical code	Ask Claude Code to pull the shared logic into one helper function both templates call, rather than maintaining two near-copies that will drift apart over time.
Library grows past a handful of effects and gets hard to navigate	Keep the README.md current as you add each template; it is the fastest way to find the right file by name instead of searching code.

Ready to go deeper?

Want the videos made for you? Paste any link or idea into Content Engine and get a published video in under 3 minutes, with DM automation armed to capture leads.

Meet Content Engine →

https://builds.digicuratoragency.com/content-engine/