Eigenflow

GUIDE ANIMATION

Le prompt de chaque style

9 styles d'animation à faire avec Claude Code

PAR

Abyane Nabeebaccus

INSTAGRAM

@abyane_nb

ÉDITION

novembre 2026

Tu as commenté ANIMATION, voici le guide. Il est pour toi si tu veux fabriquer des animations verticales pour Instagram, TikTok ou YouTube Shorts sans logiciel de montage : tu décris la scène, Claude Code écrit le code, rend la vidéo image par image et te donne un fichier MP4.

Tu y trouves le prompt de chacun des neuf styles de la vidéo, à coller avec ton sujet, puis la façon d'en faire des commandes comme /aquarelle ou /led. Chaque prompt reprend la recette de nos propres reels dans ce style, et sous chaque prompt, les pièges qu'on a rencontrés en les fabriquant.

Sources relues le 8 novembre 2026. Eigenflow n'a aucun lien avec Anthropic ni avec les auteurs des bibliothèques citées.

À savoir avant de lire

- Un prompt est un point de départ. Nos reels dans ces styles ont demandé plusieurs allers-retours et des relectures image par image. Ta première version sera moins détaillée que nos images : compte deux ou trois passes.
- On a testé un prompt tel quel, celui du panneau LED, le 8 novembre : collé dans une session Claude Code neuve, dans un dossier vide, sur le sujet « Le premier café du matin ». Il a rendu une vidéo de 10 secondes en 26 minutes, sans aide. Le résultat est en section 8. Les huit autres prompts viennent de nos recettes et n'ont pas été passés tels quels.
- Dans la vidéo, l'appli est une maquette faite en code et l'animation apparaît en une seconde. En vrai, Claude écrit les fichiers, rend les images une par une et te donne un MP4 : ça se compte en minutes. Nos animations ont été faites dans Claude Code, dans le terminal d'un Mac.
- Il te faut Claude Code (abonnement Pro, Max, Ultra, Team ou Enterprise), Node.js, ffmpeg et Google Chrome. Aucun service payant en plus, sauf pour la feutrine si tu fais générer tes personnages par un modèle d'images.
- Licences : Remotion, qui sert à cinq styles, est gratuit pour un particulier et pour une entreprise de 3 personnes au plus ; au-delà, il faut sa licence d'entreprise. three.js, p5.brush, rough.js, react-peeps et le kit Anatomy sont sous licence MIT, p5.js sous LGPL 2.1, les dessins Open Peeps dans le domaine public (CC0), les polices citées sous licence OFL (Apache 2.0 pour Permanent Marker).
- Si une marque apparaît dans ta vidéo, prends son logo officiel en SVG. Jamais un logo redessiné ou généré.

---

1. Installer une fois

Sur un Mac, ouvre le Terminal et installe Claude Code avec la commande officielle :

> curl -fsSL https://claude.ai/install.sh | bash

Ouvre une nouvelle fenêtre de Terminal, puis vérifie que la commande répond par un numéro de version :

> claude --version

Il faut un abonnement Pro, Max, Ultra, Team ou Enterprise, ou un compte Console : l'offre gratuite de claude.ai n'inclut pas Claude Code. Au premier lancement de `claude`, la connexion se fait dans le navigateur. Sous Windows, la page d'installation donne la commande PowerShell `irm https://claude.ai/install.ps1 | iex`.

Node.js fait tourner les bibliothèques de dessin. Prends la version LTS sur nodejs.org (la 26 depuis le 28 octobre 2026). Remotion demande au moins Node 16 et, sur Mac, macOS 15 ou plus récent.

> node --version

ffmpeg assemble les images en vidéo et sert aux vérifications de la partie 12. Sur Mac, avec Homebrew :

> brew install ffmpeg

Remotion embarque son propre ffmpeg, limité aux codecs vidéo dont il a besoin : installe quand même le tien pour les quatre autres styles et pour les planches. Sous Linux, ffmpeg est dans les paquets de ta distribution ; pour Windows, ffmpeg.org renvoie vers des versions compilées (gyan.dev, BtbN).

Google Chrome : l'aquarelle, la 3D de nuit, l'éclaté et le pixel art se rendent image par image dans un Chrome sans fenêtre, piloté par Puppeteer. La commande `npm i puppeteer` télécharge un Chrome compatible ; `puppeteer-core` ne télécharge rien et se sert du Chrome que tu lui indiques. Remotion cherche un Chrome installé et télécharge le sien s'il n'en trouve pas.

Crée enfin un dossier de travail et lance Claude Code dedans :

> mkdir mes-animations && cd mes-animations && git init

> claude

Au premier prompt d'un style, Claude installera ses bibliothèques avec npm et te demandera la permission. Pour les cinq styles en Remotion, l'équipe de Remotion publie aussi des skills pour les agents, en option : `npx skills add remotion-dev/skills`.

---

2. Faire de chaque prompt une commande /style

Une commande comme /aquarelle est une skill de Claude Code : un dossier qui contient un fichier SKILL.md. Le nom du dossier devient la commande. L'ancien format, un fichier `.claude/commands/aquarelle.md`, marche toujours et crée la même commande /aquarelle, mais la documentation conseille les skills pour tout ce qui est nouveau.

1. Dans ton dossier de travail, crée le dossier de la skill : `.claude/skills/aquarelle/` pour ce projet seulement, ou `~/.claude/skills/aquarelle/` pour tous tes projets sur cette machine.

2. Dedans, crée un fichier SKILL.md avec l'en-tête ci-dessous.

3. Colle sous l'en-tête le prompt du style, et remplace `[TON SUJET]` par `$ARGUMENTS`.

4. Lance `claude` dans le dossier et tape la commande suivie de ton sujet.

> mkdir -p .claude/skills/aquarelle

À COLLER

```
---
description: Vidéo verticale 1080 × 1920 en dessin animé à l'aquarelle, sur le sujet donné
argument-hint: "[sujet]"
disable-model-invocation: true
---

(colle ici le prompt du style, avec $ARGUMENTS à la place de [TON SUJET])
```

Ce que font les trois lignes de l'en-tête :

- `description` dit à Claude ce que fait la skill. C'est le seul champ que la documentation recommande.
- `argument-hint` s'affiche pendant que tu tapes la commande, pour te rappeler qu'elle attend un sujet.
- `disable-model-invocation: true` réserve la commande à toi : Claude ne la lancera pas tout seul au milieu d'une conversation. Une vidéo prend du temps et de l'abonnement, autant décider quand.

Le sujet passe en argument : tout ce que tu tapes après le nom de la commande remplace `$ARGUMENTS`. Si le fichier ne contient pas `$ARGUMENTS`, Claude Code ajoute ta phrase à la fin sous la forme `ARGUMENTS: ...`. On l'a vérifié le 8 novembre 2026 avec Claude Code 2.1.280, dans un dossier vide : la skill a reçu la phrase entière, virgules comprises.

> /aquarelle Avant les téléphones, un banc et un arrêt de bus

Fais pareil pour les huit autres styles : un dossier par commande (`nuit-3d`, `eclate`, `papier-creme`, `enquete`, `led`, `aplats`, `feutrine`, `pixel`), le prompt de la partie correspondante dedans.

Depuis l'onglet Code de l'appli Claude

Dans l'onglet Code de l'appli, une session tourne dans le cloud, sur une machine Ubuntu 24.04 qui clone ton dépôt GitHub. Elle charge les skills commitées dans le `.claude/skills/` du dépôt (ou activées sur ton compte claude.ai), jamais celles du `~/.claude/skills/` de ton Mac.

- Node 22 et Chrome y sont déjà installés, ffmpeg n'est pas dans la liste : ajoute `apt install -y ffmpeg` au script d'installation de l'environnement.
- Les limites annoncées : environ 4 processeurs, 16 Go de mémoire, et une commande attend 2 minutes par défaut (10 au plus) avant de passer en arrière-plan, où elle a encore jusqu'à 30 minutes.
- Autre voie, Remote Control : `claude remote-control` sur ton ordinateur, puis la session dans l'onglet Code. Le code tourne alors sur ta machine, avec tes fichiers et tes outils.

Nos rendus ont tourné en local sur un Mac. On n'a essayé ces commandes ni dans le cloud ni depuis le téléphone.

3. /aquarelle : un dessin animé peint

Papier grainé, lavis qui débordent un peu, trait d'encre redessiné douze fois par seconde comme une animation faite à la main, et des personnages ronds qui réagissent à tout. En haut, une étiquette de papier collée porte le titre au marqueur.



La scène /aquarelle de la vidéo, sur « Avant les téléphones ».

Quand t'en servir : une histoire avec des personnages et des objets, où l'idée se montre par une image physique (des cartons qui s'empilent, une porte qui déborde, une file d'attente).

Le prompt

> Fabrique une vidéo verticale de 10 secondes (1080 × 1920, 30 images par seconde, sans son) en dessin animé à l'aquarelle, sur ce sujet : [TON SUJET]. Outils : p5.js 2 et p5.brush dans une page HTML, rendue image par image par Chrome sans écran avec Puppeteer, puis ffmpeg (H.264, -crf 18, -pix_fmt yuv420p). Chaque image est une fonction pure du temps : aucune horloge, aucun requestAnimationFrame, hasard à graine fixe. Le look : papier grainé, lavis d'aquarelle et trait d'encre qui « bouillonnent » 12 fois par seconde (la graine du pinceau change, la position reste), des personnages ronds et simples que tu inventes (yeux, bouche, émotions qui changent, jamais immobiles), un décor où chaque objet porte une idée du sujet. En haut, entre y = 240 et y = 640, une étiquette de papier collée avec un titre en français au marqueur (Permanent Marker) et une ligne au pinceau (Caveat Brush), qui change à chaque idée. Rythme : l'image 0 montre déjà la scène complète, puis un événement toutes les 0,6 s environ (un geste, un objet qui tombe, une étiquette), la caméra dérive toujours un peu, et chaque raccord a sa propre transition (coup de pinceau, feuille qui glisse, iris, zoom dans un objet). Rien d'important sous y = 1 500 ni à droite de x = 950 entre y = 1 150 et y = 1 800. Avant de me rendre la vidéo, sors une image fixe toutes les 2 s et une planche à 2 images par seconde, relis-les, corrige ce qui est vide, coupé ou illisible, et recommence jusqu'à ce que ce soit propre. Donne-moi le chemin du MP4 et ce qui reste faible.

Les pièges qu'on a rencontrés

- Un grand lavis avec beaucoup de diffusion (le réglage bleed de p5.brush) déborde de sa forme : chez nous, une bande sombre au-dessus du sol. On l'a réglé à 0,02 sur les grands aplats.
- Une étiquette qui sort doit perdre son papier avant son texte, sinon la planche montre une étiquette vide pendant quelques images.
- Nos réglages viennent de p5.brush 2.2.3. La version 3.0.0, sortie le 28 octobre 2026, a refait les remplissages à l'aquarelle : si Claude installe la 3, revois les valeurs de diffusion à l'œil.

---

4. /nuit-3d : des personnages en 3D, la nuit

Une pièce la nuit, en 3D : fenêtre sur la ville au crépuscule, lampe chaude, léger brouillard, et un personnage fait de formes simples (sphères, cylindres) qui cligne des yeux, soupire et réagit. Aucun modèle 3D à acheter, tout est construit en code avec three.js.



La scène /nuit-3d de la vidéo, sur « Deadline, 22 h ».

Quand t'en servir : un moment vécu raconté en sketch (une deadline, une nuit blanche, une conversation avec un assistant), avec une preuve qui peut apparaître en carte au-dessus du personnage.

Le prompt

> Fabrique une vidéo verticale de 10 secondes (1080 × 1920, 30 images par seconde, sans son) en 3D de nuit, sur ce sujet : [TON SUJET]. Outils : three.js dans une page HTML, rendue image par image par Chrome sans écran avec Puppeteer, puis ffmpeg (H.264, -crf 18, -pix_fmt yuv420p). Toute la scène est une fonction pure du temps : positions, poses, clignements et caméra calculés depuis t, aucune horloge. Le look : une pièce la nuit, une fenêtre sur la ville au crépuscule, une lampe chaude qui découpe la lumière, un léger brouillard. Le personnage est fait de primitives : tête en ellipsoïde, cheveux en calotte, yeux avec paupières en demi-sphères qui clignent, sourcils en arcs de tore, bouche en trait courbe, bras en cylindres avec rotules, et quelques poses qu'on mélange (repos, réflexion, épaules qui tombent). Pas de vernis brillant sur la peau. Caméra : un plan par idée, poussée lente, coup de zoom sec sur le moment fort, personnage à environ 1,5 m de l'objectif. Rythme : l'image 0 montre déjà la scène complète, un événement toutes les 0,6 s environ, rien d'immobile plus de 0,5 s. Titre de contexte en français entre y = 240 et y = 640 ; rien d'important sous y = 1 500 ni à droite de x = 950 entre y = 1 150 et y = 1 800. Avant le rendu complet, projette les positions clés à l'écran et sors une image fixe toutes les 2 s : vérifie que la caméra n'entre dans aucun objet, que les yeux s'ouvrent vraiment et que rien ne cache le visage. Puis planche à 2 images par seconde, corrections, et donne-moi le chemin du MP4 et ce qui reste faible.

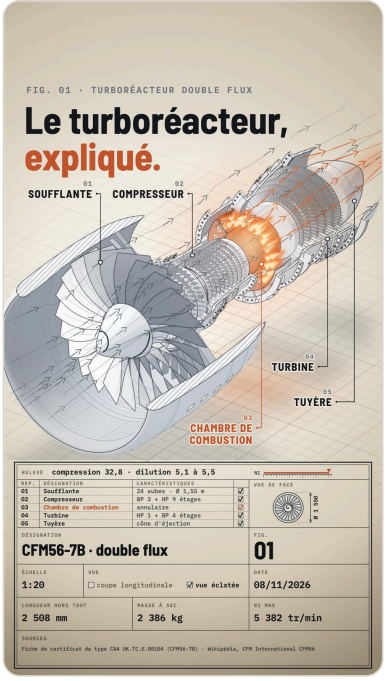
Les pièges qu'on a rencontrés

- `CapsuleGeometry(1, 1)` de three.js mesure 3 unités de haut une fois les deux bouts comptés : nos premiers bras étaient démesurés. Un cylindre d'une unité plus deux rotules donne la bonne taille.
- Une paupière en demi-sphère, à 0 radian, couvre déjà la moitié de l'œil : regard endormi. Chez nous, l'œil ouvert est à -0,95 rad pour la paupière du haut et +1,0 pour celle du bas, et l'iris reste sous le rayon de la paupière, sinon il la perce quand l'œil se ferme.
- En 9:16, on ne voit qu'une vingtaine de degrés de ciel au-dessus de l'horizon : les couleurs du crépuscule doivent descendre jusqu'à 10°, sinon elles restent hors du cadre.

---

5. /eclate : une machine qu'on démonte

Un dessin technique en perspective isométrique, au trait graphite sur papier quadrillé : la machine du sujet se démonte pièce par pièce, chaque pièce porte son étiquette, une seule couleur attire l'œil. La caméra passe d'une pièce à l'autre dans le même dessin.



La scène /eclate de la vidéo, sur « Le turboréacteur, expliqué ».

Quand t'en servir : expliquer comment marche un objet ou un outil (un moteur, une appli, une chaîne d'outils branchés entre eux), quand les entrées, le mécanisme et le résultat peuvent devenir des pièces.

La skill Anatomy (licence MIT) sait déjà construire ces figures isométriques. Elle est en option, et s'installe ainsi d'après son dépôt :

> npx skills add wheresryan22/anatomy -g -a claude-code

Le prompt

> Fabrique une vidéo verticale de 10 secondes (1080 × 1920, 30 images par seconde, sans son) en vue éclatée isométrique, sur ce sujet : [TON SUJET]. Outils : une fonction JavaScript draw(état) qui renvoie un SVG de 1080 × 1920, sans dépendance (ou avec la skill Anatomy si elle est installée), rendue image par image par Chrome sans écran avec Puppeteer, puis ffmpeg (H.264, -crf 18, -pix_fmt yuv420p). L'état est une fonction pure du temps. Le look : dessin technique au trait graphite sur papier quadrillé, ombrage en quatre tons, une seule couleur d'accent, la lumière d'une lampe. L'objet est l'explication : la machine du sujet, construite en dizaines de petites pièces précises, se démonte le long de ses axes, et chaque pièce reçoit une étiquette reliée par un filet de cote (Barlow Semi Condensed pour les titres, IBM Plex Mono pour les étiquettes). Une seule scène : la caméra est un zoom 2D du SVG, traits en vector-effect: non-scaling-stroke pour rester nets à tout zoom, qui passe d'une pièce à l'autre avec une lente avancée. Les pièces flottent et rebondissent un peu en continu. Respecte l'ordre de dessin, du fond vers l'avant. Rythme : l'image 0 montre déjà la machine complète, un geste toutes les 0,6 s environ (une pièce qui s'écarte, une étiquette, un flux qui traverse). Titre en français entre y = 240 et y = 640, avec un fondu de papier derrière lui ; rien d'important sous y = 1 500 ni à droite de x = 950 entre y = 1 150 et y = 1 800. Pour la capture, fenêtre exactement à 1080 × 1920 et aucune option clip. Avant de me rendre la vidéo : une image fixe toutes les 2 s et une planche à 2 images par seconde, vérifie qu'aucune pièce n'en cache une autre ni une étiquette, corrige, recommence. Donne-moi le chemin du MP4 et ce qui reste faible.

Les pièges qu'on a rencontrés

- `page.screenshot({ clip })` force Chrome à recalculer toute la page : 6 s par image au lieu de 40 ms. Avec une fenêtre à la taille exacte et `captureBeyondViewport: false`, notre reel entier se rendait en une minute et demie.
- Un objet posé devant un autre sur la même diagonale le masque à l'écran : place les pièces en regardant leur projection, pas le plan vu de dessus.
- Le texte d'un écran vu en biais déborde vite. Compte les caractères avant d'écrire (chez nous, environ 3 unités par caractère à 5 px).

---

6. /papier-creme : un croquis sur papier crème

Une page de carnet crème : le texte s'écrit mot à mot, les mots clés passent en italique rouge, des encadrés sont entourés ou barrés au crayon, des tampons claquent, et un petit personnage dessiné réagit en bas de la page.



La scène /papier-creme de la vidéo, sur « Ta vie de fondateur ».

Quand t'en servir : une liste d'erreurs, de pièges ou d'étapes chiffrées, ou une journée racontée en croquis.

Le prompt

> Fabrique une vidéo verticale de 10 secondes (1080 × 1920, 30 images par seconde, sans son) en croquis sur papier crème, sur ce sujet : [TON SUJET]. Outils : Remotion (React), rough.js avec une graine fixe pour les cercles, ratures et tampons au crayon, react-peeps pour le personnage, polices Instrument Sans et Instrument Serif. Tout se calcule depuis l'image courante (useCurrentFrame), rien sur une horloge. Le look : papier crème tramé ; en haut, « // 01 · » et le nom de la partie, un compteur et une barre à crans qui avance ; le texte s'écrit mot à mot, mots clés en italique rouge ; au milieu, un encadré par idée (une note, un chiffre, un objet), entouré ou barré au crayon, avec un tampon ; en bas, un personnage Open Peeps, la tête au-dessus de y = 1 500, qui change d'expression sur les mots clés et se penche à chaque phrase ; les grands chiffres arrivent dans un halo rouge tramé. Une partie passe sur fond rouge, une autre sur fond noir, avec une transition en trame entre les deux. Rythme : l'image 0 montre déjà la page complète, un événement toutes les 0,6 s environ, entrées en 5 images, tampons en 3. Titre entre y = 240 et y = 640 ; rien d'important sous y = 1 500 ni à droite de x = 950 entre y = 1 150 et y = 1 800. Avant de me rendre la vidéo, sors une image fixe toutes les 2 s (npx remotion still) et une planche à 2 images par seconde, relis, corrige, recommence. Rends avec npx remotion render et donne-moi le chemin du MP4 et ce qui reste faible.

Dans la scène de la vidéo, le personnage est l'avatar dessiné d'Abyane, en deux poses générées par GPT Image. Le prompt reprend Open Peeps, dessiné en code, comme notre premier reel dans ce style.

Les pièges qu'on a rencontrés

- Pose tous les mots d'une phrase dès le départ, invisibles, puis révèle-les un par un. Sinon la phrase se recompose pendant qu'elle s'écrit et les mots sautent.
- Un tampon posé sur un dessin hachuré disparaît : donne-lui un fond blanc à 90 %.
- Un chiffre rouge sur la partie à fond rouge disparaît : passe-le en blanc avec un halo rouge sombre.

---

7. /enquete : le tableau d'enquête de nuit

Un mur de liège la nuit, des dossiers punaisés, un fil rouge qui relie les preuves, et une caméra qui saute de pièce en pièce comme dans une série policière. Les documents sont réels, surlignés au feutre.



La scène /enquete de la vidéo, sur « 404 av. J.-C., Sparte ».

Quand t'en servir : un sujet à preuves (une affaire, une arnaque, une règle de droit, une histoire avec des dates et des lieux).

Le prompt

> Fabrique une vidéo verticale de 10 secondes (1080 × 1920, 30 images par seconde, sans son) en tableau d'enquête de nuit, sur ce sujet : [TON SUJET]. Outils : Remotion (React), une texture de liège raccordable générée en code et posée en dalles d'images (pas en fond CSS), des documents réels du domaine public ou des captures de pages publiques, avec la source et la date en petit sur chaque pièce. Tout se calcule depuis l'image courante. Le look : un seul plan, un mur de liège la nuit avec des dossiers punaisés en grille (deux colonnes, cinq rangs, pour remplir le 9:16 au recul), un fil rouge qui relie les preuves en passant sous les papiers (punaises au-dessus), des passages surlignés, un feutre qui entoure, des tampons usés, des chiffres à rouleaux, des post-it, des feuilles qui balancent depuis leur punaise, une lampe qui s'allume au recul. Caméra { x, y, zoom, rotation } qui recadre sur chaque mot clé en 15 à 21 images, coups de zoom légers, petites secousses ; quand elle file vite, mélange l'image avec les deux précédentes pour un flou de mouvement. Rythme : l'image 0 montre déjà le mur complet, un événement toutes les 0,6 s environ. Titre de contexte en français entre y = 240 et y = 640, sur une bande sombre ; rien d'important sous y = 1 500 ni à droite de x = 950 entre y = 1 150 et y = 1 800. Avant de me rendre la vidéo : une image fixe toutes les 2 s et une planche à 2 images par seconde, vérifie que le fil ne barre aucun texte et que chaque document se lit, corrige, recommence. Donne-moi le chemin du MP4 et ce qui reste faible.

Les pièges qu'on a rencontrés

- Posé au-dessus des feuilles, le fil rouge barrait les documents. Il passe dessous, seules les punaises restent au-dessus.
- D'après la documentation de Remotion, une image posée en `background-image` CSS n'est pas attendue au rendu et peut clignoter. Notre liège passe par des composants `Img` en dalles, que Remotion attend.
- Notre premier jet était « un poil trop rapide » et « un tout petit peu raide » : on a allongé les mouvements de caméra de 11 à 15 images à 15 à 21, et réduit les coups de zoom à 60 % et les secousses à 50 %.

---

8. /led : le panneau de points lumineux

Un panneau de 90 × 160 points lumineux sur fond noir, rouges et blancs, où tout se dessine en points : mots géants, chiffres, objets. Des titres nets se posent par-dessus, et le panneau pivote un peu à chaque mot.



La scène /led de la vidéo, sur « Ouvrir un cœur d'IA ».

Quand t'en servir : une question du type « qu'est-ce qui se passe si… », un chiffre fort, une annonce. Chez nous, c'est le style le plus vu : médiane de 15 230 vues sur ses reels, contre 2 088 pour l'ensemble du compte (Buffer, 4 novembre 2026).

Le prompt

> Fabrique une vidéo verticale de 10 secondes (1080 × 1920, 30 images par seconde, sans son) en panneau LED, sur ce sujet : [TON SUJET]. Outils : Remotion (React). Le panneau : 90 × 160 points lumineux sur fond noir, chacun avec un petit halo. Dessine la scène dans un calque de 360 × 640 en mode additif (canal rouge pour les points rouges, canal vert pour les points blancs à gain bas), réduis-le à 90 × 160, puis allume chaque point selon sa valeur. Traits et cadres d'au moins 8 px du calque, sinon un point ne s'allume qu'à moitié. Les chiffres et les mots en points passent par une police de panneau 5 × 7 calée sur la grille, un point de police pour une LED. Par-dessus, des titres nets en Inter, mot clé en rouge. Le panneau pivote légèrement à chaque mot (perspective, rotateX, rotateY), les mots géants entrent en 3 ou 4 images avec trois échos en traînée, et une ligne de balayage rouge fait passer d'une scène à l'autre. Tout se calcule depuis l'image courante. Rythme : l'image 0 montre déjà le panneau allumé, un événement toutes les 0,6 s environ, aucun passage de plus de 1,6 s sans nouvel élément. Titre entre y = 240 et y = 640 ; rien d'important sous y = 1 500 ni à droite de x = 950 entre y = 1 150 et y = 1 800. Avant de me rendre la vidéo : une image fixe toutes les 2 s et une planche à 2 images par seconde, vérifie que chaque chiffre en points se lit juste et qu'aucun mot géant ne perd ses bords, corrige, recommence. Donne-moi le chemin du MP4 et ce qui reste faible.

Ce prompt, testé tel quel

Le 8 novembre 2026, on a collé ce prompt sans rien changer, sujet « Le premier café du matin », dans une session Claude Code neuve (Opus 5.5, aucun réglage perso, dossier vide). Claude a monté le projet Remotion, rendu, relu ses propres planches et corrigé seul : 26 minutes, 43 tours. La vidéo fait 10 secondes et tient le rythme demandé, un événement toutes les 0,3 à 0,5 seconde : le réveil passe de 06:59 à 07:00, la mouture monte à 18 g, l'eau à 92 °C coule 25 secondes, la batterie remonte à 100 %, avec les mots géants DRING, CRRRK, PSHHH et ENFIN. C'est plus simple que nos scènes, mais c'est une base propre, que tu peux enrichir en deux ou trois passes.



Le résultat du test : cinq images de la vidéo rendue par Claude Code avec ce seul prompt, sujet « Le premier café du matin » (8 novembre 2026).

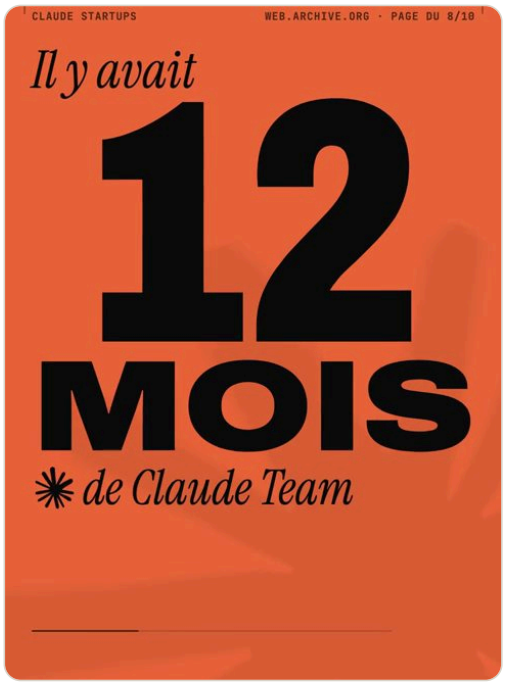
Les pièges qu'on a rencontrés

- Une photo non détourée s'allume en bloc rouge. Détoure d'abord, et éclaircis de 30 % les objets sombres, sinon ils disparaissent dans le noir.
- Un trait de 3 à 5 px du calque n'allume qu'un quart de point une fois réduit : traits et cadres à 8 px, soit deux points.
- Sous 40 unités du calque, une police ordinaire réduite rend les chiffres faux : « 150 » se lisait « 180 ». La police de panneau 5 × 7 règle le problème.

---

9. /aplats : des aplats et des lettres géantes

Chaque scène a sa couleur plein cadre, le texte passe d'un mot énorme à une page dense, les chiffres roulent comme une machine à sous, et on passe d'une scène à l'autre en traversant le trou d'une lettre.



Une image de notre reel 79 : « 1 000 $ de crédits » en lettres géantes sur un aplat bleu.

Quand t'en servir : une annonce, une offre, un chiffre à faire retenir.

Le prompt

> Fabrique une vidéo verticale de 10 secondes (1080 × 1920, 30 images par seconde, sans son) en aplats et lettres géantes, sur ce sujet : [TON SUJET]. Outils : Remotion (React), et la police Archivo (OFL) exportée lettre par lettre en chemins SVG avec leurs contreformes, en Black large et Black étroit (avec fontTools), pour que chaque mot remplisse la largeur sans déformation. Chaque scène a sa couleur plein cadre. Le texte passe de l'énorme au minuscule : une scène n'a qu'un mot, la suivante est dense. Les nombres s'empilent pour remplir l'écran (45 sur une ligne, 000 € sur la suivante) et roulent comme une machine à sous, chiffre par chiffre. Chaque plan naît du précédent : on traverse le trou d'une lettre ou d'un 0, le clip-path suit le vrai tracé, et la scène suivante est déjà là dans le trou. Un logo officiel, jamais redessiné, peut grandir jusqu'à devenir le fond. Dans les coins, de minuscules mentions en police à chasse fixe (source, date) donnent l'échelle. Aucun fondu. Tout se calcule depuis le temps en secondes : rends à 240 images par seconde, puis passe ffmpeg -vf "tmix=frames=6,fps=30" pour un vrai flou de mouvement, et encode en H.264 (-crf 18, -pix_fmt yuv420p). Rythme : l'image 0 est déjà pleine, un événement toutes les 0,6 s environ, aucun aplat vide plus de 0,5 s. Rien d'important sous y = 1 500 ni à droite de x = 950 entre y = 1 150 et y = 1 800. Avant le rendu complet : une image fixe toutes les 2 s à 30 images par seconde, une planche à 2 images par seconde, corrections. Donne-moi le chemin du MP4 et ce qui reste faible.

Les pièges qu'on a rencontrés

- Un chiffre en machine à sous s'affiche sur sa valeur finale avant de rouler si on ne le découpe pas chiffre par chiffre, chacun avec son départ décalé de 0,045 s.
- À 120 images par seconde avant le mélange, des images doubles se voyaient sur les mouvements rapides. On a gardé 240 images par seconde avec `tmix=frames=6`.
- Une légende de la même couleur que le grand numéro plein écran devient illisible : coupe-la tant que le numéro remplit l'écran.

---

10. /feutrine : une cuisine en feutrine

Un décor en feutrine filmé en stop-motion : des personnages en tissu qui bougent par à-coups, des étiquettes et des bons en papier, des tampons, une jauge en feutre qui se fait croquer.



Une image de notre reel 82 : le chef en feutrine dans sa cuisine, la jauge de limite en bas.

Quand t'en servir : expliquer un usage par des rôles (le chef, le commis, le second), une astuce de méthode.

Les personnages et le décor de notre reel sont des images générées par GPT Image, puis détourées ; tout le reste est en code. Le prompt prend tes images dans un dossier `assets/`, et sinon dessine une version de secours en SVG, qu'on n'a pas testée.

Le prompt

> Fabrique une vidéo verticale de 10 secondes (1080 × 1920, 30 images par seconde, sans son) en stop-motion de feutrine, sur ce sujet : [TON SUJET]. Outils : Remotion (React). Les personnages et le décor sont les images PNG détourées du dossier assets/ (un personnage en plusieurs poses, un décor en 9:16, des objets). S'il n'y en a pas, dessine des personnages simples en SVG avec une texture de feutre (bruit, couture en pointillés) et dis-moi que c'est une version de secours. Tout ce qui porte du texte se fait en code : étiquettes, bons en papier au bas en dents de scie, jauge en feutre, tampons qui claquent, avec une texture de feutre posée par-dessus en mix-blend-mode: multiply, chargée par le composant Img de Remotion (pas en background-image, qui peut clignoter au rendu), et une couture en pointillés. Les personnages bougent à 12 images par seconde avec un léger tremblé de stop-motion ; la caméra et les objets bougent à 30. Entre deux scènes denses, un aplat plein cadre d'une seule couleur avec un mot. Titre de contexte en français entre y = 240 et y = 640, dès la première image, dessiné après les objets pour qu'aucun ne passe devant. Tout se calcule depuis l'image courante. Rythme : l'image 0 montre déjà la scène complète, un événement toutes les 0,6 s environ, aucune scène qui commence vide plus de 0,5 s. Rien d'important sous y = 1 500 ni à droite de x = 950 entre y = 1 150 et y = 1 800. Avant de me rendre la vidéo : une image fixe toutes les 2 s et une planche à 2 images par seconde, relis, corrige, recommence. Donne-moi le chemin du MP4 et ce qui reste faible.

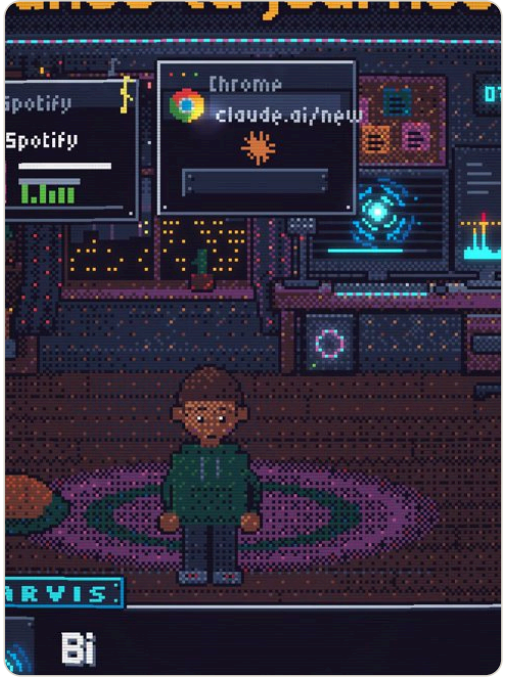
Les pièges qu'on a rencontrés

- Le signe moins de la police Bodoni Moda est un simple filet : nos tampons et nos chiffres sont passés en Archivo noir.
- Un objet qui entre par le haut traverse le titre. Dessine le titre après les objets, ou fais apparaître l'objet sur place.
- Une fiche affichée à l'écran doit être un extrait exact du fichier qu'elle montre, avec « (extrait) » dans son titre.

---

11. /pixel : un petit jeu en pixel art

Un petit jeu vidéo en pixel art : une image de 216 × 384 pixels agrandie cinq fois, 32 couleurs, des ombres tramées, une boîte de dialogue de jeu de rôle et un léger effet d'écran cathodique.



Une image de notre reel 54 : la chambre en pixel art et les fenêtres que l'assistant ouvre.

Quand t'en servir : un outil ou un dépôt GitHub montré comme une quête en étapes, avec les écrans du test en gros plan.

Le prompt

> Fabrique une vidéo verticale de 10 secondes (1080 × 1920, 30 images par seconde, sans son) en pixel art, sur ce sujet : [TON SUJET], montré comme un petit jeu vidéo. Outils : un canvas 2D sans dépendance de 216 × 384 pixels, agrandi 5 fois sans lissage, rendu image par image par Chrome sans écran avec Puppeteer, puis ffmpeg (H.264, -crf 18, -pix_fmt yuv420p). Palette Endesga 32 seulement. Chaque matière a une rampe de quatre ou cinq couleurs de la palette, ombrée par un tramage ordonné (Bayer 4 × 4) ; quelques lumières ponctuelles ; un contour d'un pixel autour des personnages ; une passe d'écran cathodique légère en WebGL (lignes de balayage, bords à peine arrondis) à la toute fin. Texte seulement en polices à grille stricte, à leur taille native (Press Start 2P ou Silkscreen à 8 px), jamais entre deux tailles. Mise en page : titre de contexte en haut, la scène au milieu, une boîte de dialogue de jeu de rôle pour les phrases, au-dessus de y = 1 500 une fois agrandie. Transitions en iris, en trame ou en stores. Tout se calcule depuis l'image courante. Rythme : l'image 0 montre déjà la scène complète, un événement toutes les 0,6 s environ (un personnage qui marche, une fenêtre qui s'ouvre, un objet qui brille). Rien d'important à droite de x = 950 entre y = 1 150 et y = 1 800 une fois agrandie. Avant de me rendre la vidéo : une image fixe toutes les 2 s en taille réelle et une planche à 2 images par seconde, vérifie chaque lettre et chaque chiffre, corrige, recommence. Donne-moi le chemin du MP4 et ce qui reste faible.

Les pièges qu'on a rencontrés

- Pixelify Sans casse une fois agrandie : elle n'est pas dessinée sur une grille stricte. On s'est limité aux polices qui le sont.
- Chrome pose les lettres à un demi-pixel près. Notre parade : rendre chaque lettre 8 fois plus grande, puis lire le centre de chaque case. Sans ça, un « e » devenait un « o ».
- Un numéro de ligne faux à l'écran (386 au lieu de 387) n'a été vu qu'après coup : vérifie chaque nombre affiché contre son fichier source.

---

12. Rendre et vérifier

Les prompts demandent déjà à Claude de relire ses images. Relis-les aussi : plusieurs pièges de ce guide, on les a vus sur une planche, pas en regardant la vidéo.

La planche à 2 images par seconde

Cette commande prend 2 images par seconde, trace en rouge la colonne des boutons d'Instagram et en jaune la zone du pseudo et de la légende, puis range 40 images par planche, soit 20 secondes de vidéo. On l'a testée le 8 novembre 2026 avec ffmpeg 8.0.1.

> cd mes-animations && ffmpeg -i video.mp4 -vf "fps=2,drawbox=x=950:y=1150:w=130:h=650:color=red@0.6:t=8,drawbox=x=0:y=1500:w=1080:h=420:color=yellow@0.5:t=8,scale=270:-1,tile=8x5" planche-%02d.jpg

Pour regarder un instant précis en pleine taille, ici 1,2 s :

> cd mes-animations && ffmpeg -ss 1.2 -i video.mp4 -frames:v 1 image-1-2s.png

Les trous dans le rythme

Un passage de plus de 1,6 s où l'image change à peine se voit. Cette commande les liste :

> cd mes-animations && ffmpeg -i video.mp4 -vf "select='gt(scene,0.004)',metadata=print:file=-" -an -f null - 2>/dev/null | grep -o "pts_time:[0-9.]*" | cut -d: -f2 | awk 'NR>1 && $1-p>1.6 {printf "trou de %.1f s avant %.1f s\n", $1-p, $1} {p=$1}'

Elle ne voit pas les mouvements continus et très doux (un compteur, un reflet) : tranche à la planche avant de corriger.

Les zones à laisser libres

Nos repères sur un 1080 × 1920, mesurés sur nos reels publiés :

- le haut, jusqu'à y = 220 : rien d'important ;
- les titres entre y = 240 et y = 640 ;
- sous y = 1 500 : le pseudo et la légende recouvrent l'image ;
- la colonne de droite, x > 950 entre y = 1 150 et y = 1 800 : ni texte ni chiffre, les boutons j'aime, commentaire et partage sont dessus.

Assembler des images en MP4

Les styles rendus par Chrome sortent des images numérotées. Pour les assembler :

> cd mes-animations && ffmpeg -framerate 30 -i frames/%05d.png -c:v libx264 -crf 18 -pix_fmt yuv420p -movflags +faststart video.mp4

D'après le wiki de ffmpeg, sans `-pix_fmt yuv420p` la vidéo peut ne pas se lire dans QuickTime et la plupart des lecteurs. Même chemin pour les styles en Remotion : le 8 novembre, un MP4 rendu directement par Remotion est sorti chez nous en yuvj420p, avec une piste audio muette. On rend donc en images PNG (`npx remotion render` avec `--sequence --image-format=png`), puis on assemble avec cette commande. La documentation de Remotion donne aussi les options `--muted` et `--pixel-format=yuv420p`, qu'on n'a pas essayées.

Sources

Pages relues le 8 novembre 2026.

- Claude Code : skills et commandes, installation, sessions dans le cloud, environnements cloud, appli mobile, Remote Control.
- Remotion : licence et prix, LICENSE.md, démarrage et prérequis, skills pour agents, ffmpeg intégré depuis la v4.
- Bibliothèques : three.js, p5.js, p5.brush, rough.js, react-peeps, Open Peeps, Anatomy, Puppeteer.
- Outils : Node.js, calendrier des versions de Node.js, ffmpeg, ffmpeg sur Homebrew, wiki ffmpeg sur H.264, palette Endesga 32, Google Fonts.