

YANIS CADI · STUDIO

CINEMATIC 3D WEBSITES

The complete method to build a website where the camera travels through the place as you scroll, like in a film.
From the first images to the live site.

01 THE METHOD

02 READY-TO-USE TOOLS

03 SELLING

THIS GUIDE IS MY OPEN WORKSHOP

You have seen my videos: a website you visit like a film, the camera moving down a hallway, into a screen, diving into a café all the way to the cup. Many of you asked me how I do it. Here is everything.

This guide is not theory. It is the method I use for my own websites, with tools I wrote and tested on real projects. Every command in this guide was run and checked before being printed here.

You will learn to build the effect from A to Z: write the camera's journey, generate shots that fit perfectly inside each other, edit them into one continuous film, and make it play with the scroll on a web page that loads fast, on desktop and on mobile.

WHAT YOU WILL BE ABLE TO DO

- ▶ Design a "Russian doll" storyboard
- ▶ Generate consistent shots with AI
- ▶ Upscale your images to high definition
- ▶ Edit the film with my script
- ▶ Build the page that plays with the scroll
- ▶ Put it online and sell the result

WHAT YOU NEED

- ▶ A computer (Mac, Windows or Linux)
- ▶ An image generator that accepts a reference image (ChatGPT, for example)
- ▶ Python 3 and ffmpeg, both free
- ▶ A free Netlify account
- ▶ One weekend for your first site

NO CODING SKILLS REQUIRED

You do not need to know how to program. The three tools come ready to use: you copy them, run one command, and get your film and your page. I still explain what each important line does, so you can adapt it.

THE JOURNEY, SHOT BY SHOT

01	WHAT A CINEMATIC 3D WEBSITE IS The effect, why it works, what it is not	p. 04
02	THE RUSSIAN DOLL PRINCIPLE The golden rule that makes a continuous zoom possible	p. 07
03	WRITING THE STORYBOARD From the wide shot to the detail that makes people want to come	p. 10
04	GENERATING THE SHOTS WITH AI Prompts, consistency, in-between shots	p. 13
05	WORKING WITH REAL PHOTOS Clients, people, likeness, honesty	p. 17
06	HIGH DEFINITION Upscaling without blur or a plastic look	p. 20
07	EDITING THE FILM My film_zoom.py script, step by step	p. 23
08	ENCODING FOR THE WEB Smooth on scroll, light to load	p. 27
09	THE PAGE THAT PLAYS THE FILM My page_scroll.html template explained	p. 30
10	GOING LIVE Netlify drag and drop, domain name	p. 33
11	THE TRAPS I PAID FOR Eight mistakes and how to avoid them	p. 34
12	SELLING 3D WEBSITES Who to, how to present, what price	p. 37
+	APPENDICES Checklist, prompts, glossary, full code	p. 40

A 3D rendered scene of a tropical cafe. The cafe has a thatched roof and is surrounded by palm trees. Inside, there is a counter with a coffee machine and various items. The scene is dimly lit, with warm lights from the interior and hanging lamps. The overall atmosphere is cozy and inviting.

01

WHAT A CINEMATIC 3D WEBSITE IS

A website you do not browse: you step into it. Every turn of the mouse wheel moves the camera forward through the place.

THE SCROLL BECOMES A CAMERA

On a classic website, you scroll through blocks of text and photos. On a cinematic 3D website, the scroll drives a camera moving through one continuous scene.



My opening film: from the hallway to the desk, into the screen, then into the café all the way to the cup. One single camera move, driven by the scroll.

Technically, it is a **video** whose displayed frame depends on the scroll position. When the visitor scrolls down 10 %, the video moves forward 10 %. When they scroll back up, it moves back. They feel like they are driving the camera themselves.

This video is not filmed. It is built from a series of images that fit inside each other: each image is a detail of the previous one. By zooming gradually from one to the next, you get a continuous tracking shot, with no cut, that gives a real feeling of depth. That is what we call "3D" here: depth you can feel, not a 3D model.

KEY TAKEAWAY

Images that fit inside each other → a continuous zoom film → a video driven by the scroll. The whole guide follows these three steps.

THREE REASONS PEOPLE STAY

01

CURIOSITY

The visitor wants to see what comes next. Every scroll reveals a new place: they go all the way down to find out where the camera is taking them.

02

MEMORY

People remember a journey, not a grid of photos. The place is told in the order you would discover it in real life.

03

STANDING OUT

Almost no local business has this. The site gets shared, shown and talked about: it becomes a selling point in itself.

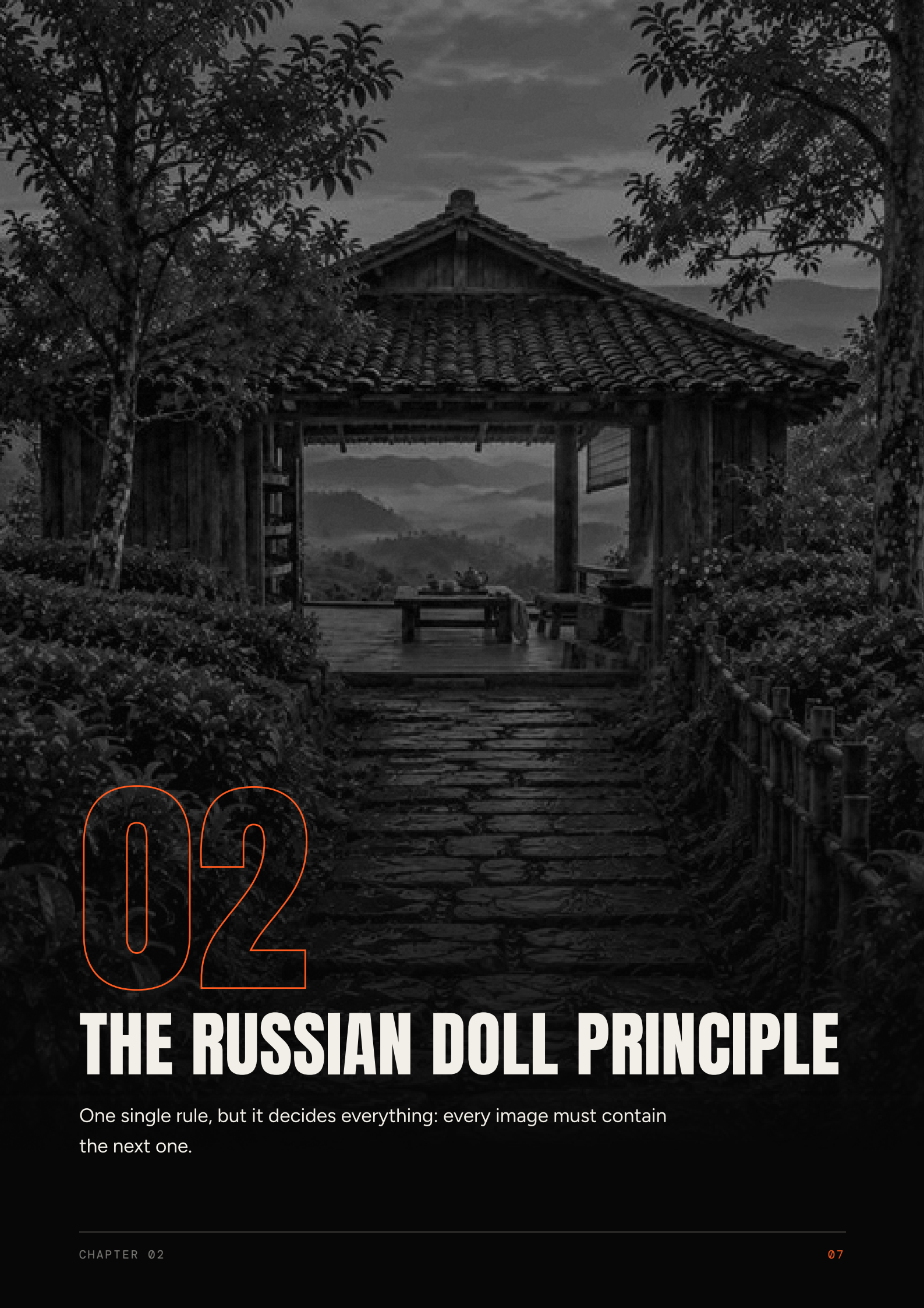
► WHICH PLACES?

The effect works best when there is a **path** to tell: you arrive, you go in, you sit down, you discover the product.

TYPE OF PLACE	THE CAMERA'S JOURNEY
Café, restaurant	The street or the beach, the front, the dining room, the counter, the dish or the cup
Hotel, guesthouse	The landscape, the entrance, the lounge, the room, the view from the bed
Shop, brand	Where the product comes from, the workshop, the hands that make it, the finished object
Craftsperson	The work site, the tool, the gesture, the finishing detail

WHAT IT IS NOT

It is not a real-time 3D video game, and it is not 3D modelling software. No need for Blender or a powerful graphics card: everything relies on images and a video. That is what makes the method fast, light, and accessible without being a developer.



02

THE RUSSIAN DOLL PRINCIPLE

One single rule, but it decides everything: every image must contain the next one.

EVERY SHOT CONTAINS THE NEXT ONE

Think of Russian dolls. The biggest one contains the middle one, which contains the small one. Your film works the same way: shot 2 is an area of shot 1, shot 3 is an area of shot 2.



The camera starts on shot 1 and zooms towards the area where shot 2 sits. Just before it gets there, shot 2, which is more detailed, takes over with a crossfade. The zoom continues, and so on. The viewer never sees a cut: they see a camera moving forward.

▶ THE THREE CONDITIONS

1. **Same point of view.** The next shot is taken in the direction the camera is moving. No sudden change of angle.
2. **Same light, same time of day, same weather.** A sunset cannot turn into bright midday in the next shot.
3. **A reasonable zoom.** Between two shots, the detail should fill between 60 % and 80 % of the width of the previous shot. Beyond that, the jump shows.

THE NUMBER TO REMEMBER

My best transitions have a scale of **0.60 to 0.80**. My script shows it for every transition: if you see 0.40, an in-between shot is missing (chapter 04).

A REAL SEQUENCE, BROKEN DOWN



Six shots from my Mộc Sương website, in order. Each image is the enlarged and enriched centre of the previous one.

Look closely: the pavilion in the background of shot 1 becomes the subject of shot 2, and the stone path of shot 2 leads to the door of shot 3. Nothing is left to chance. The subject of the next shot is already **visible, in the centre**, of the current one.



The same opening edited by my script: 12 frames taken every 10 frames of the film. The camera moves forward without a cut.



03

WRITING THE STORYBOARD

Before generating a single image, write the journey. Ten minutes of storyboard save you two hours of failed shots.

FROM THE WIDEST TO THE MOST DESIRABLE

A good journey starts far away and ends on what the customer came for: the cup, the dish, the room, the object.

01 THE OPENING SHOT

Wide, beautiful, recognisable. It sets the scene: the beach, the valley, the street. It is also the image shown before the video loads, so it must be strong on its own.

02 THE APPROACH

Two to four shots to get closer: the front, the entrance, the doorstep. This is where the visitor understands they are about to go in.

03 THE INSIDE

Two to four shots inside the place: the room, the counter, the workshop, the gestures. This is the heart of its identity.

04 THE FINAL SCENE

The product, in close-up, perfect. The steaming cup, the dish, the unmade bed facing the sea. It is the image that makes people want to come, and it is where you place your call-to-action button.

HOW MANY SHOTS?

6 to 8 shots for a first site. **12 to 14** for a very smooth result, by adding in-between shots. My opening film has 13, for a 36-second journey.

THE STORYBOARD OF A BEACH CAFÉ

Here is the real storyboard of my Kafe Talay demo. Copy this table for your project: one row per shot, the subject in the centre, and what must already be visible to prepare the next shot.

SHOT	WHAT YOU SEE	ALREADY VISIBLE IN THE CENTRE
01	The beach at sunset, palm trees, the café in the distance	The café roof
02	The wooden front, the boardwalk across the sand	The entrance under the roof
03	Under the roof, the tables, the lamps switched on	The counter
04	The counter, the coffee machine, the jars	A cup on the counter
05	The cup, latte art, coffee beans, kraft bag	End of the journey

► THE TITLES ALONG THE JOURNEY

Each stage of the journey gets a short title that appears during the scroll. Write them at the same time as the storyboard:

MOMENT	TITLE	ROLE
Shot 01	"Come closer."	Promise, invitation to scroll
Shots 02 to 03	"Step inside."	The experience of the place
Shot 04	"Make yourself at home."	The lived moment
Shot 05	"Book your table."	The call to action



04

GENERATING THE SHOTS WITH AI

The secret is not the perfect prompt. It is always starting from the previous image.

ALWAYS START FROM THE PREVIOUS SHOT

If you generate each shot from scratch, you get beautiful images that do not connect. Consistency comes from one single habit: giving the previous shot as the reference image.

01 GENERATE THE OPENING SHOT

Landscape 16:9 format, as large as possible. Describe the place, the time of day, the light, the photo style. Keep this prompt: you will reuse its words in every following shot.

```
# Prompt for shot 01
Realistic photo, landscape 16:9 format. A small wooden beach café
by the water, at sunset, palm trees, white sand, warm golden light
coming from the right. Wide view from the beach, the café in the
centre of the image, at mid-distance. High-end travel photography
style, no text, no logo, no people.
```

02 GENERATE EACH NEXT SHOT FROM THE PREVIOUS ONE

Provide the previous image and explicitly ask to move towards the centre, keeping everything else identical.

```
# Prompt for the following shots (with the previous image attached)
Here is the previous shot of my film. Generate the next shot:
the camera has moved straight forward towards the centre of the
image, about 1.4 times closer. We now see [the café front, the
boardwalk]. Keep exactly the same light, the same time of day, the
same colours, the same photo style, 16:9 format. No text.
```

THE TRICK THAT CHANGES EVERYTHING

Always say "the camera has moved straight forward towards the centre". If the tool moves the subject to the side, the transition becomes difficult: regenerate rather than keeping an off-centre image.

WHEN THE JUMP IS TOO BIG

Between two very different shots, the beach and the cup for example, the zoom is too strong: the crossfade shows. The solution is an in-between shot, made with my 79 % crop technique.

1. **Crop the centre of shot N** to 79 % of its width and height, then enlarge this crop back to the original size. You get a slightly blurry but perfectly placed image: this is your **geometric reference**.
2. **Give the AI this reference and shot N+1**, and ask for an image exactly halfway: the framing of the reference, with sharp and faithful details.
3. **Name it shot N-bis** (01a, 02a...) and place it between the two.

```
# 79 % crop with ffmpeg (one line)
ffmpeg -i plan-02.png -vf "crop=iw*0.79:ih*0.79,scale=1672:941" reference-02a.png
```

```
# Prompt for the in-between shot (attach reference-02a.png and plan-03.png)
Image 1: exact framing to respect (crop of the previous shot).
Image 2: the shot the camera is moving towards.
Generate a sharp photo with exactly the framing of image 1,
adding the realistic details you would discover as you get
closer, faithful to image 2. Same light, 16:9 format.
```

► QUALITY CHECK FOR EACH SHOT

- ☐ The subject of the next shot is visible and centred
- ☐ No invented text, no real brand logo
- ☐ Hands and faces without defects (five fingers, natural gaze)
- ☐ Same time of day and same light colour as the previous shot

A CLEAN FOLDER, A CLEAN FILM

My script reads the images in alphabetical order. From the start, use a numbering that never breaks the order:

```
mon-projet/  
  plans/  
    01.png    the beach  
    01a.png   in-between shot added later  
    02.png    the café front  
    03.png    under the roof  
    04.png    the counter  
    05.png    the cup  
  hd/        the upscaled versions (chapter 06)  
  site/      the page and the videos (chapters 08 to 10)
```

"01a" naturally sorts between "01" and "02". So you can add an in-between shot at any time without renaming the rest.

RECOMMENDED FORMAT

PNG, **16:9**, at least **1672 × 941** pixels. My script crops automatically if an image is not exactly the right format, but it is better to generate in 16:9 directly.

KEEP YOUR PROMPTS

Save every prompt in a text file next to the images. When a client asks for a change six months later, you regenerate in two minutes instead of reinventing everything.

05

WORKING WITH REAL PHOTOS

A website for a real business must look like the real business. AI enhances, it does not invent.

ENHANCE WITHOUT LYING

Your client welcomes visitors who have seen the website. If they find a smaller room or a different dish, trust is gone. The rule: improve the light and the framing, never add anything that does not exist.

01 START FROM THE CLIENT'S PHOTOS

Ask for four photos: the outside, the inside, a dish or a drink, the view. Use them as reference images for each matching shot, with this instruction:

Here is a real photo of the place. Generate a brighter, better framed version, in 16:9 format, keeping exactly the same furniture, colours, materials and objects. Do not add anything that is not in the photo.

02 PEOPLE

If a person appears in the film (the owner, or yourself), two rules:

- ▶ **Their written consent** before anything is published.
- ▶ **Likeness is achieved with a local edit.** First generate the scene, then edit only the face, providing two or three real photos of the person as reference. A face generated "at random" is smooth and generic: friends and family will not recognise them.

MY EXPERIENCE

On my own film, the first generated face did not look like me. Editing the face alone, with my photos as reference, changed everything: beard, eye shape, jawline. The rest of the image stayed identical, down to the pixel.

FIX A DETAIL WITHOUT BREAKING EVERYTHING

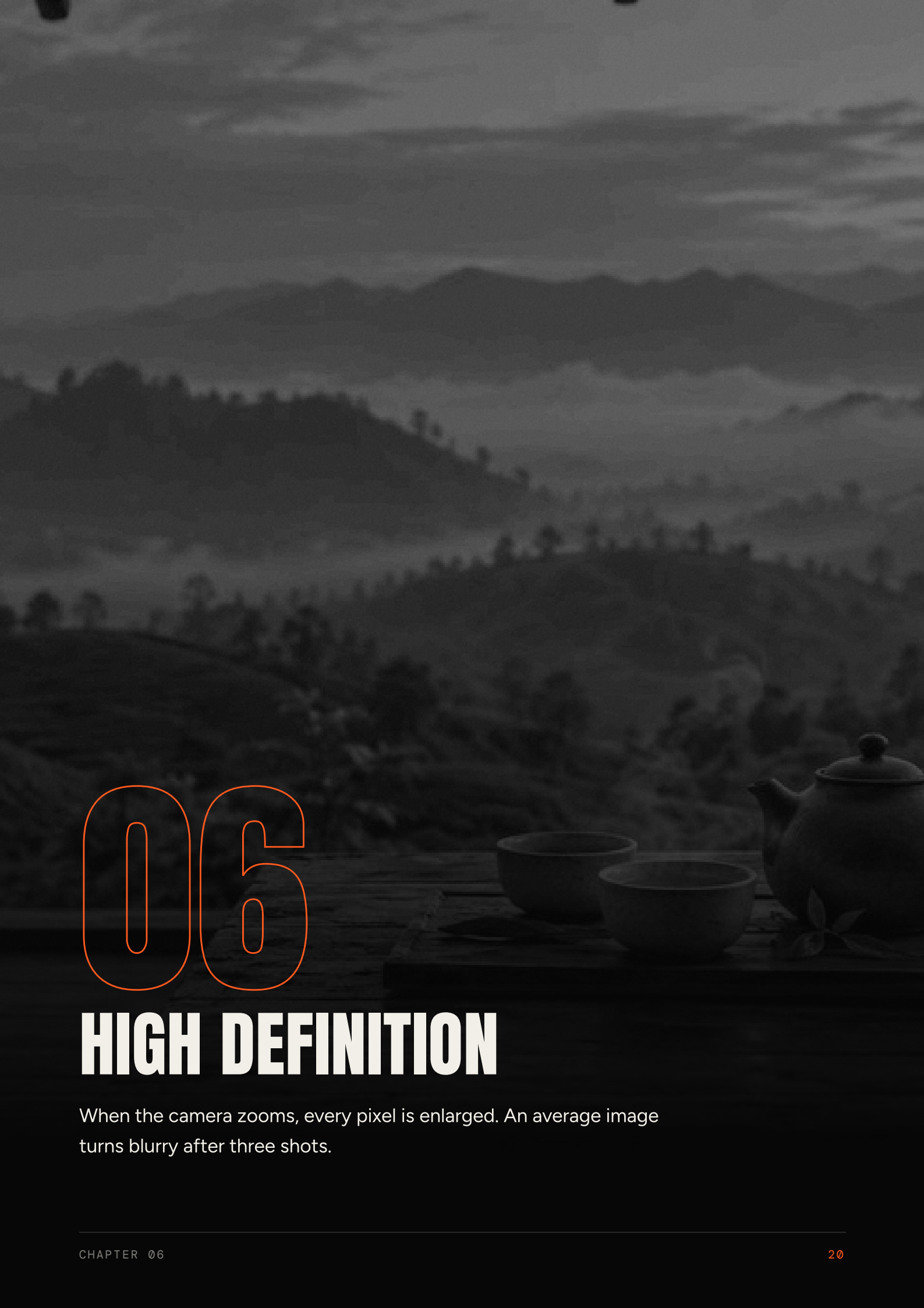
A shot is almost perfect, but one detail is wrong: a TV instead of a computer screen, a hand with six fingers, an invented logo. Do not regenerate the whole shot, you would lose the transition with the others. Make a local edit.

1. **Crop the area to fix** with a generous margin around it.
2. **Edit this crop** with AI, describing only the fix.
3. **Paste the area back** into the original image with a soft edge.
4. **Check** that nothing moved outside the area: that is what keeps the transitions with the neighbouring shots perfect.

```
# Local edit prompt
Edit only [the screen on the desk]: replace it with a thin black
bezel computer monitor showing a beach café website. Everything
else in the image must stay strictly identical: light,
perspective, objects, people.
```

► ANIMATING A GESTURE

Same principle to make a hand move: generate two or three variants where **only the hand changes** (open, half closed, closed), then alternate them during the film. Two slow gestures are better than twenty fast ones: a gesture repeated too often looks mechanical.

The background of the page is a dark, atmospheric photograph. It shows a misty mountain landscape with layers of hills and valleys. In the foreground, on the right side, there is a wooden tray holding a traditional tea set, including a teapot and two cups. The overall tone is dark and moody, with a focus on texture and depth.

06

HIGH DEFINITION

When the camera zooms, every pixel is enlarged. An average image turns blurry after three shots.

DOUBLE THE RESOLUTION OF EVERY SHOT

During a transition, the camera sometimes shows only 60 % of the image. On a wide screen, those 60 % are enlarged: that is where blur appears. The fix is to work with shots twice as large as the final screen.

01 INSTALL REAL-ESRGAN (FREE)

Download the version for your system (Mac, Windows or Linux) from the releases page of the Real-ESRGAN project on GitHub (file `realesrgan-ncnn-vulkan`), then unzip it. No installation, no account.

02 UPSCALE A SHOT

```
# x4 upscale with the realistic photo model
./realesrgan-ncnn-vulkan -i plans/02.png -o hd/02.png -n realesrgan-x4plus
```

03 GO BACK TO X2 FOR A NATURAL LOOK

The raw x4 sometimes looks "plastic". My recipe: shrink the x4 to twice the original size, blend it at 60 % with a classic upscale, and add a very light photo grain. The result is sharp without looking artificial.

```
# Bring the x4 back to x2 (example for a 1672 × 941 shot)
ffmpeg -i hd/02.png -vf "scale=3344:1882:flags=area" hd/02-x2.png
```

WHY NOT X4 DIRECTLY?

A x4 file 6688 pixels wide slows down the whole edit for an invisible gain. The x2 (about 3300 pixels) is enough for a perfect result in 1080p and even beyond.

WHAT UPSCALING DOES NOT DO

Upscaling makes the image sharper. It does not make it more faithful. If a face is 50 pixels wide in the original shot, upscaling it will give a sharp face... that is still approximate.

WHAT UPSCALING DOES

- ▶ Sharp edges
- ▶ Believable textures (wood, foliage, fabric)
- ▶ A film that stays clean in full screen

WHAT IT DOES NOT DO

- ▶ Make a face look like the real person
- ▶ Fix a badly drawn hand
- ▶ Repair unreadable text

The order of operations matters:

1. Shots generated and approved
2. x2 upscale
3. Local edits on the upscaled versions (face, hands), with the reference photos
4. Film editing

Editing after upscaling lets you work on the face in high definition: the AI has more pixels to draw a three-day beard, a gaze, real skin.



07

EDITING THE FILM

My `film_zoom.py` script assembles your shots into one continuous tracking shot. One command.

PREPARE YOUR COMPUTER

You only do this once. Allow ten minutes.

01 PYTHON 3

On Mac, Python 3 is often already there: type `python3 --version` in the Terminal. Otherwise, download it from python.org. On Windows, tick "Add Python to PATH" during installation.

02 FFMPEG

The tool that makes the videos. On Mac with Homebrew: `brew install ffmpeg`. On Windows: download a build from ffmpeg.org and add its folder to the PATH.

03 THE SCRIPT'S TWO LIBRARIES

```
pip install opencv-python numpy
```

04 CHECK

```
python3 -c "import cv2, numpy; print('OK')"  
ffmpeg -version
```

If both commands answer without an error, you are ready.

WHERE TO FIND THE TOOLS

The three files (`film_zoom.py` , `encoder_web.sh` , `page_scroll.html`) come with this guide in the "outils" (tools) folder. Their full code is also printed in the appendix, so you do not depend on anything. Their comments and messages are in French: the commands work exactly the same.

ONE COMMAND, ONE FILM

```
python3 film_zoom.py hd film.mp4
```

The script reads all the images in the `hd` folder in order, works out for each transition where the next shot sits in the current one, then builds the film. It prints a report (in French, as below):

```
01.webp → 02.webp : échelle 0.61, centre (0.49, 0.51), calé automatiquement
02.webp → 03.webp : échelle 0.70, centre (0.50, 0.51), calé automatiquement
03.webp → 04.webp : échelle 0.64, centre (0.49, 0.46), calé automatiquement
Film prêt : film.mp4
```

▶ READING THE REPORT

YOU SEE	IT MEANS	WHAT TO DO
calé automatiquement (aligned automatically), échelle (scale) 0.55 to 0.85	The script found the next shot inside the current one	Nothing, it is perfect
zoom central par défaut (default centre zoom)	The two shots are too different to be aligned	Normal when changing rooms. Otherwise, add an in-between shot
échelle (scale) below 0.5	The jump is too big	Add an in-between shot (chapter 04)

▶ USEFUL SETTINGS

OPTION	EFFECT
<code>--images-par-raccord 90</code>	Length of each transition, in frames. 90 frames = 3 seconds at 30 frames per second
<code>--zoom 0.78</code>	Zoom used when automatic alignment fails
<code>--largeur 1920 --hauteur 1080</code>	Film size (width, height). Keep 16:9

HOW THE SCRIPT CREATES THE ILLUSION

You do not need to understand these details to use it, but they will help you diagnose a transition you do not like.

01 ALIGNMENT

The script spots thousands of feature points (corners, textures) in both shots and works out how to overlay the next shot onto the current one. From that it deduces a **scale** and a **centre**.

02 EXPONENTIAL ZOOM

The camera does not zoom at a constant speed in pixels, but in proportion: $\text{zoom} = \text{scale}^t$. That is what gives a feeling of steady forward motion, like a real tracking shot, instead of a zoom that speeds up.

03 THE ANCHORED CROSSFADE

In the second half of the transition, the next shot fades in gradually, placed exactly where it belongs, with soft edges. When the camera arrives, it fills the whole screen: you have moved from one shot to the next with no visible cut.

04 THE FINAL PAUSE

The film ends with a one-second pause on the last shot: the visitor has time to see the final scene before the text and the call-to-action button.

CHECKING SMOOTHNESS

Open the film and drag the playhead slowly. If you see a "jump" at a transition, it is almost always an off-centre shot: regenerate it with the subject in the centre, or add an in-between shot.



08

ENCODING FOR THE WEB

A stunning film that stutters on scroll is useless. Encoding makes all the difference.

SMOOTH ON SCROLL, LIGHT TO LOAD

A normal video is optimised to be played from start to finish. You want the browser to jump anywhere, instantly, in both directions. That is set at the encoding stage.

```
bash encoder_web.sh film.mp4
```

The script produces four files: the desktop video (1920 × 1080), the mobile video (1080 × 1920, vertical), and their two poster images.

▶ WHAT MATTERS IN THE COMMAND

SETTING	WHY
<code>-g 8</code>	One keyframe every 8 frames. The browser can seek anywhere without decoding long sequences. This is the most important setting.
<code>-bf 0</code>	No "bidirectional" frames: scrolling back stays instant.
<code>-crf 26</code>	The balance between quality and weight. 22 = sharper and heavier, 30 = lighter and softer.
<code>-movflags +faststart</code>	The video starts playing before it is fully downloaded.
<code>-an</code>	No sound: a muted video can start on its own on mobile.

MY WEIGHT BENCHMARK

For a 36-second journey in 1080p: about **25 MB** on desktop and **20 MB** on mobile, at very high quality. If your film is longer, raise the `crf` by 2 points rather than lowering the resolution.

ENCODER_WEB.SH

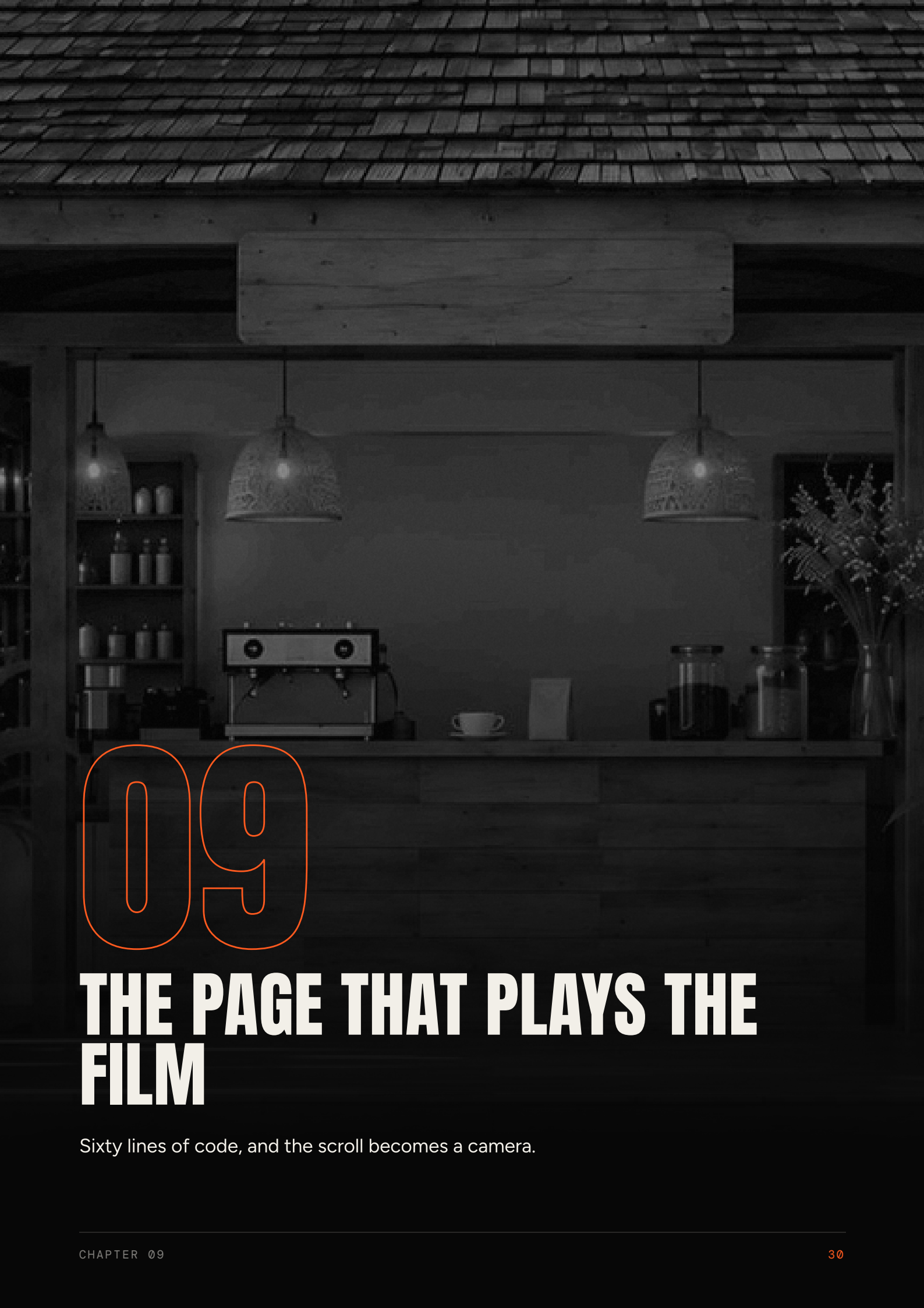
```
#!/bin/bash
# encoder_web.sh · prépare ton film pour le web (défilement fluide au scroll)
# Utilisation : bash encoder_web.sh film.mp4
# Crée film-web.mp4 (ordinateur, 1920x1080) et film-mobile.mp4 (téléphone, 1080x1920)
set -e
IN="$1"; BASE="${IN%.*}"
# Une image clé toutes les 8 images : le navigateur peut sauter partout dans la vidéo sans
# ramer.
ffmpeg -y -i "$IN" -an -vf "scale=1920:1080:flags=lanczos" -c:v libx264 -preset slow -tune
film -crf 26 -g 8 -bf 0 -pix_fmt yuv420p -movflags +faststart "$BASE-web.mp4"
# Version téléphone : on garde le centre de l'image au format vertical.
ffmpeg -y -i "$IN" -an -vf "crop=ih*9/16:ih,scale=1080:1920:flags=lanczos" -c:v libx264 -
preset slow -tune film -crf 26 -g 8 -bf 0 -pix_fmt yuv420p -movflags +faststart "$BASE-
mobile.mp4"
# Image d'attente affichée avant le chargement de la vidéo.
ffmpeg -y -i "$BASE-web.mp4" -frames:v 1 -q:v 3 "$BASE-poster.jpg"
ffmpeg -y -i "$BASE-mobile.mp4" -frames:v 1 -q:v 3 "$BASE-poster-mobile.jpg"
echo "Prêt : $BASE-web.mp4, $BASE-mobile.mp4 et les deux images d'attente."
```

► THE MOBILE VERSION

On a phone held upright, a 16:9 film would shrink to a thin strip. So the script crops the centre of the image to a vertical format. If your subject is not in the centre in the first shots (a person on the left, for example), adjust the crop position:

```
# Crop at 34 % from the left instead of the centre
-vf "crop=ih*9/16:ih:(iw-ih*9/16)*0.34:0,scale=1080:1920"
```

For my opening film, the crop follows the person on the left during the first third, then gradually returns to the centre as the camera enters the screen. It is an advanced setting, but this detail is what makes the mobile version as beautiful as the desktop one.



09

THE PAGE THAT PLAYS THE FILM

Sixty lines of code, and the scroll becomes a camera.

PAGE_SCROLL.HTML, IN THREE IDEAS

Put `page_scroll.html` (renamed `index.html`) in the same folder as your videos and poster images. Open it: your site already works.

01 A SCENE STUCK TO THE SCREEN

The `.film` block is very tall (`400vh` , four screens). Inside it, the scene "sticks" to the top of the screen (`position: sticky`) while you scroll through those four screens. The height sets the length of the journey: the taller it is, the more you scroll.

02 THE SCROLL BECOMES VIDEO TIME

```
function avancement(){
  var h = film.offsetHeight - innerHeight;
  return Math.min(1, Math.max(0, -film.getBoundingClientRect().top / h));
}
cible = avancement() * (video.duration - .05);
```

At 0 % scroll, the start of the video is shown, at 100 % the end. In between, everything is proportional.

03 THE CAMERA SMOOTHLY CATCHES UP WITH THE SCROLL

```
if (Math.abs(cible - actuel) > .008) {
  actuel += (cible - actuel) * .2;
  video.currentTime = actuel;
}
```

Instead of jumping straight to the right frame, the video covers 20 % of the remaining distance at each screen refresh. This small easing gives the feeling of a real camera move, even with a jerky mouse wheel.

TITLES, MOBILE, IPHONE

▶ THE TITLES THAT APPEAR

```
<div class="titre" data-de=".3" data-a=".6"><h2>Entrez</h2></div>
```

Each title has a display window: **data-de** (from) and **data-a** (to), between 0 (start of the journey) and 1 (end). Replace the text with your own titles, then place them by watching your film: at what percentage do you reach the counter? Put the title just before.

▶ DESKTOP OR MOBILE

```
var mobile = innerWidth / innerHeight < .8;
video.src = mobile ? 'film-mobile.mp4' : 'film-web.mp4';
```

A screen taller than it is wide gets the vertical version. Each visitor downloads only one video: the page stays light.

▶ THE IPHONE DETAIL

Safari on iPhone refuses to seek a video that has never been played. So the template plays it for a few milliseconds on the first touch of the screen, then pauses it. Without this detail, the film stays frozen on iPhone.

THE REST OF THE PAGE

After the film, it is a normal page: introduction, menu, opening hours, map, contact. Keep a **call-to-action button visible from the final scene** of the film: that is when desire is strongest.

LIVE IN FIVE MINUTES

Netlify hosts sites like yours for free, with a secure https address. No server to set up.

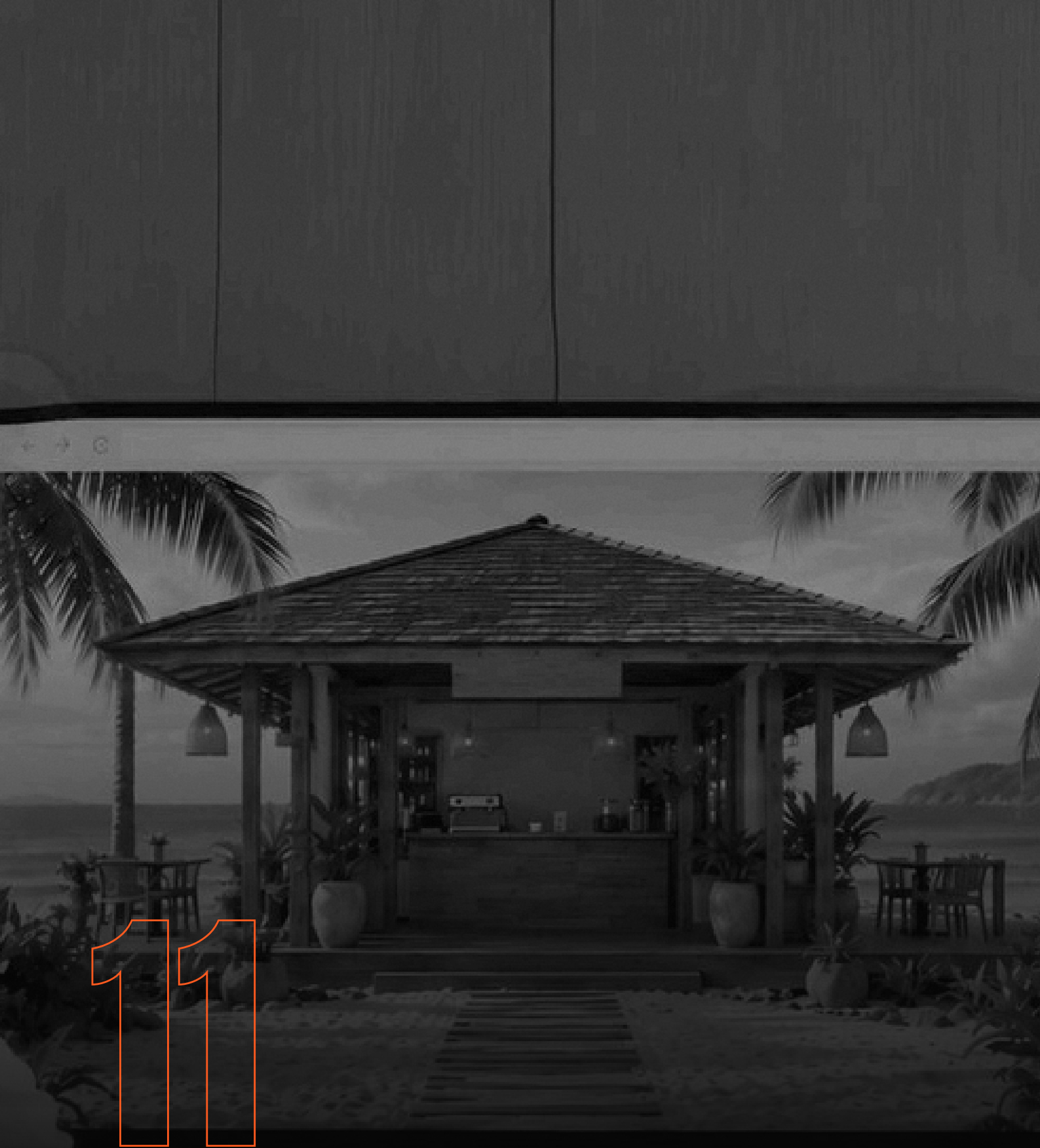
1. Gather in a `site` folder: `index.html`, `film-web.mp4`, `film-mobile.mp4`, `film-poster.jpg`, `film-poster-mobile.jpg`.
2. Create a free account on netlify.com.
3. Open the "Netlify Drop" page and **drag and drop the `site` folder**.
4. Your site is live at an address like `random-name.netlify.app`. You can rename it in the site settings.

▶ A REAL DOMAIN NAME

For a client, buy their domain (around 10 € a year from a registrar such as OVH, Gandi or Namecheap), then add it in Netlify under "Domain management". Netlify shows you the two DNS settings to make at the registrar and creates the https certificate automatically.

▶ CHECK BEFORE SENDING TO THE CLIENT

- ☐ On desktop: the film follows the scroll in both directions
- ☐ On a real iPhone and a real Android phone, not only in a simulator
- ☐ On 4G, not only on Wi-Fi: the poster image shows up immediately
- ☐ The titles appear at the right moment
- ☐ The call-to-action button works (phone, WhatsApp or form)



11

THE TRAPS I PAID FOR

Eight mistakes I made on real projects, so you do not make them.

EIGHT MISTAKES, EIGHT FIXES

THE PROBLEM	THE FIX
Straight lines that wobble (posts, window frames) during a transition	Happens when you force a warp between two shots that are too different. My script does a clean zoom and crossfade, with no warping: keep it as it is.
The ghost: a half-transparent person during a crossfade	Two shots where the person changes position (standing, then sitting). Shorten this transition or make the person disappear one shot earlier.
Blur in full screen	The video was downsized to save weight. Encode at 1920 wide minimum and adjust the <code>cif</code> , not the resolution.
The film stutters on scroll	Video encoded without <code>-g 8 -bf 0</code> . Re-encode with <code>encoder_web.sh</code> .
The generic face	Local edit of the face alone, on the upscaled version, with real photos as reference (chapters 05 and 06).
Invented text on a sign or a screen	Write "no readable text" in every prompt, and check every shot in full screen before editing.
The person cut off on mobile	The vertical crop takes the centre. Shift it towards your subject (chapter 08).
Changing one shot shifts everything	If you replace a shot, check the length of the neighbouring transitions: a better aligned transition can lengthen or shorten part of the journey and shift your titles.

WATCH YOUR FILM FRAME BY FRAME BEFORE YOU SHOW IT.

Almost every defect you have just read about shows up in three seconds when you step through the video slowly, and almost never at normal speed. Clients scroll slowly: they will see it.

► MY CHECKING ROUTINE

1. **Contact sheet:** one frame per second, side by side. Lighting inconsistencies jump out.
2. **Slow playback** of each transition, stepping frame by frame.
3. **Full screen on a large monitor** to spot blur.
4. **A real phone** for the vertical framing and smoothness.
5. **A fresh pair of eyes:** show it to someone without explaining anything and watch where they frown.

```
# Contact sheet: one frame per second, 6 columns  
ffmpeg -i film.mp4 -vf "fps=1,scale=320:-1,tile=6x6" -frames:v 1 planche.jpg
```



12

SELLING 3D WEBSITES

You know how to make them. Here is how to offer them to the people who need them.

SHOW BEFORE YOU SELL

A cinematic 3D website sells in ten seconds of video, not in ten lines of description. Your best weapon is a demo that moves.



Three of my demos: a tea brand in Vietnam, a mountain café in Thailand, a kebab restaurant. Three worlds, the same method.

01 CHOOSE YOUR TARGETS

Businesses where the **atmosphere** sells: cafés, restaurants, hotels, guesthouses, designer shops, tourist spots. At first, avoid businesses where customers mostly look for a price or a phone number.

02 PREPARE A DEMO MADE FOR THEM

A five-shot storyboard using the business's public photos, a short film, a page online. You show up with **their** place in motion: nothing compares to that against a generic presentation.

03 SHOW THE VIDEO OF YOUR HAND SCROLLING

Film your screen while you scroll through the site. This video works everywhere: private message, social media, meetings. It is what triggers the "I want the same thing".

SET YOUR PRICE, FILL YOUR CALENDAR

▶ WHAT PRICE?

A cinematic 3D website takes more work than a classic showcase website, and it gets noticed much more. As a benchmark, my cinematic website offer is **590 € as a one-time payment**, with no subscription. Adapt it to your market, your experience and what you include (number of shots, languages, edits, domain name).

WHAT I INCLUDE IN AN OFFER

- ▶ The journey as a film, made from photos of the place
- ▶ The desktop and mobile versions
- ▶ Menu, opening hours, map, contact
- ▶ Two languages if the customers are international
- ▶ Going live and connecting the domain name

▶ THE FUNNEL THAT TURNS A VIDEO INTO A MEETING

1. **The video** shows your hand scrolling through a site, with on screen: "Comment 3D".
2. **The comment** triggers an automatic private message (with a tool such as ManyChat).
3. **The message** checks that the person follows you, then sends your page.
4. **The page** offers two choices: book a call to have their site made, or learn to make it themselves.

This is exactly the path that brought you to this guide.

THE FULL CHECKLIST

01 PREPARE

- ☐ Storyboard written, from wide to desirable
- ☐ Subject of each shot visible in the centre of the previous one
- ☐ Journey titles written
- ☐ Client photos collected (4 minimum)
- ☐ Written consent from the people filmed

02 GENERATE

- ☐ Shot 01 in 16:9, large format
- ☐ Each shot generated from the previous one
- ☐ Same light and same time of day everywhere
- ☐ No invented text, no logo
- ☐ Hands and faces checked
- ☐ In-between shots where the jump is big

03 UPSCALE AND EDIT

- ☐ All shots upscaled x2
- ☐ Faces edited with the reference photos
- ☐ Nothing moved outside the edited areas

04 EDIT THE FILM

- ☐ film_zoom.py run, report read
- ☐ Scales between 0.55 and 0.85
- ☐ Film checked frame by frame
- ☐ Contact sheet with no break in the lighting

05 PUBLISH

- ☐ encoder_web.sh run
- ☐ Mobile crop checked
- ☐ Titles timed to the film
- ☐ Call-to-action button on the final scene
- ☐ Tested on a real iPhone and a real Android
- ☐ Tested on 4G
- ☐ Domain name connected, https active

06 SELL

- ☐ Video of your hand scrolling
- ☐ Demo in the prospect's name
- ☐ Clear offer, price shown

READY-TO-COPY PROMPTS

01 OPENING SHOT

Realistic photo, landscape 16:9 format. [The place] seen from afar, [the time of day and the light]. The subject in the centre of the image, at mid-distance. High-end [travel / architecture / food] photography style. No text, no logo, no people.

02 NEXT SHOT

Here is the previous shot of my film. Generate the next shot: the camera has moved straight forward towards the centre, about 1.4 times closer. We now see [new subject]. Same light, same time of day, same colours, same style, 16:9 format. No text.

03 IN-BETWEEN SHOT

Image 1: exact framing to respect. Image 2: the shot the camera is moving towards. Generate a sharp photo with the framing of image 1 and details faithful to image 2. Same light, 16:9 format.

04 FROM A REAL PHOTO

Here is a real photo of the place. Generate a brighter, better framed version, 16:9 format, keeping exactly the same furniture, colours, materials and objects. Do not add anything that is missing.

05 LOCAL EDIT

Edit only [the area]. [The fix]. Everything else in the image stays strictly identical: light, perspective, objects.

SHORT GLOSSARY

Shot	A still image of your journey. The film links 6 to 14 of them.
Transition	The move from one shot to the next: zoom, then crossfade.
Scale	The size of the next shot inside the current one. 0.70 = it fills 70 % of the width.
In-between shot	A shot added between two shots that are too far apart, to soften the zoom.
Local edit	Changing a single area of an image without touching the rest.
Upscaling	Increasing the resolution of an image with a specialised AI.
Keyframe	A complete image inside a video. The more frequent they are, the faster you can move around in the video.
crf	The quality setting of the encoding. The lower it is, the sharper and heavier the video.
Sticky	A page element that stays stuck to the screen while you scroll.
Poster image	The image shown before the video has loaded.
Netlify	A host that puts a website online for free with drag and drop.

FILM_ZOOM.PY

```
#!/usr/bin/env python3
"""film_zoom.py · transforme une série d'images en film à zoom continu (effet « poupée russe »).

Chaque image doit être un détail de la précédente : la caméra plonge dans l'image 1,
arrive sur l'image 2, plonge dedans, et ainsi de suite.

Installation (une seule fois) :
    pip install opencv-python numpy
    et ffmpeg (https://ffmpeg.org, ou « brew install ffmpeg » sur Mac)

Utilisation :
    python3 film_zoom.py dossier_images film.mp4
    python3 film_zoom.py dossier_images film.mp4 --images-par-raccord 90 --largeur 1920

Les images sont prises dans l'ordre alphabétique : nomme-les 01.png, 02.png, 03.png...
"""
import argparse
import subprocess
import sys
from pathlib import Path

import cv2
import numpy as np

def charger(dossier, largeur, hauteur):
    fichiers = sorted(p for p in Path(dossier).iterdir() if p.suffix.lower() in {".png", ".jpg", ".jpeg", ".webp"})
    if len(fichiers) < 2:
        sys.exit("Il faut au moins deux images dans le dossier.")
    images = []
    for p in fichiers:
        im = cv2.imread(str(p))
        h, w = im.shape[:2]
        # recadrage au format de la vidéo, centré
        cible = largeur / hauteur
        if w / h > cible:
            nw = int(h * cible); im = im[:, (w - nw) // 2:(w - nw) // 2 + nw]
        else:
            nh = int(w / cible); im = im[(h - nh) // 2:(h - nh) // 2 + nh]
        images.append(cv2.resize(im, (largeur, hauteur), interpolation=cv2.INTER_AREA))
    return fichiers, images

def trouver_cadrage(a, b, default):
    """Où se trouve l'image b à l'intérieur de l'image a ? Renvoie (échelle, centre x, centre y) en proportions."""
    orb = cv2.ORB_create(4000)
    ka, da = orb.detectAndCompute(cv2.cvtColor(a, cv2.COLOR_BGR2GRAY), None)
    kb, db = orb.detectAndCompute(cv2.cvtColor(b, cv2.COLOR_BGR2GRAY), None)
    if da is not None and db is not None and len(ka) > 20 and len(kb) > 20:
        paires = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True).match(db, da)
        if len(paires) >= 30:
            pb = np.float32([kb[m.queryIdx].pt for m in paires])
            pa = np.float32([ka[m.trainIdx].pt for m in paires])
            M, ok = cv2.estimateAffinePartial2D(pb, pa, method=cv2.RANSAC, ransacReprojThreshold=4)
            if M is not None and ok.sum() >= 40:
                s = float(np.hypot(M[0, 0], M[1, 0]))
                h, w = a.shape[:2]
                cx, cy = M @ np.array([w / 2, h / 2, 1.0])
                if 0.25 < s < 0.95:
                    return s, cx / w, cy / h, "calé automatiquement"
    return default, 0.5, 0.5, "zoom central par défaut"

def lisse(x):
    x = min(max(x, 0.0), 1.0)
    return x * x * (3 - 2 * x)
```

FILM_ZOOM.PY (CONTINUED)

```
def raccord(a, b, s, cx, cy, n, sortie):
    h, w = a.shape[:2]
    bx, by = cx * w, cy * h
    # masque aux bords adoucis pour poser b dans a sans couture
    masque = np.zeros((h, w), np.float32)
    marge = int(0.08 * min(w, h))
    masque[marge:-marge, marge:-marge] = 1
    masque = cv2.GaussianBlur(masque, (0, 0), marge / 2)[..., None]
    for i in range(n):
        t = i / n
        z = s ** t                                # zoom exponentiel : vitesse perçue constante
        k = (1 - z) / (1 - s)                    # 0 au départ, 1 à l'arrivée
        vx, vy = w / 2 + (bx - w / 2) * k, h / 2 + (by - h / 2) * k
        # vue actuelle dans a
        Ma = np.float32([[1 / z, 0, w / 2 - vx / z], [0, 1 / z, h / 2 - vy / z]])
        vue = cv2.warpAffine(a, Ma, (w, h), flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REFLECT)
        # b posé à sa place, visible de plus en plus nettement sur la fin du raccord
        alpha = lisse((t - 0.45) / 0.5)
        if alpha > 0:
            f = s / z
            Mb = np.float32([[f, 0, w / 2 + (bx - vx) / z - f * w / 2], [0, f, h / 2 + (by - vy) / z - f * h /
2]])
            pose = cv2.warpAffine(b, Mb, (w, h), flags=cv2.INTER_CUBIC)
            m = cv2.warpAffine(masque, Mb, (w, h))[..., None] * alpha
            vue = (vue * (1 - m) + pose * m).astype(np.uint8)
        sortie.write(vue.tobytes())

def main():
    ap = argparse.ArgumentParser(description="Film à zoom continu à partir d'une série d'images.")
    ap.add_argument("dossier"); ap.add_argument("film")
    ap.add_argument("--images-par-raccord", type=int, default=90, help="90 = 3 secondes à 30 images/s")
    ap.add_argument("--largeur", type=int, default=1920); ap.add_argument("--hauteur", type=int, default=1080)
    ap.add_argument("--zoom", type=float, default=0.78, help="zoom utilisé quand le calage automatique échoue")
    ap.add_argument("--qualite", type=int, default=18, help="18 = très haute qualité (master)")
    o = ap.parse_args()
    fichiers, images = charger(o.dossier, o.largeur, o.hauteur)
    ff = subprocess.Popen(["ffmpeg", "-loglevel", "error", "-y", "-f", "rawvideo", "-pix_fmt", "bgr24",
        "-s", f"{o.largeur}x{o.hauteur}", "-r", "30", "-i", "-",
        "-c:v", "libx264", "-preset", "slow", "-crf", str(o.qualite), "-pix_fmt", "yuv420p",
        "-movflags", "+faststart", o.film], stdin=subprocess.PIPE)
    for i in range(len(images) - 1):
        s, cx, cy, info = trouver_cadrag(e(images[i], images[i + 1], o.zoom)
        print(f"{fichiers[i].name} → {fichiers[i + 1].name} : échelle {s:.2f}, centre ({cx:.2f}, {cy:.2f}),
{info}")
        raccord(images[i], images[i + 1], s, cx, cy, o.images_par_raccord, ff.stdin)
    for _ in range(30):
        # une seconde de pause sur la dernière image
        ff.stdin.write(images[-1].tobytes())
    ff.stdin.close(); ff.wait()
    print("Film prêt :", o.film)

if __name__ == "__main__":
    main()
```

PAGE_SCROLL.HTML

```

<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Mon site cinématique</title>
  <style>
    *{box-sizing:border-box;margin:0;padding:0}
    body{background:#050505;color:#F2EFE8;font-family:system-ui,sans-serif}
    /* La hauteur de .film règle la longueur du voyage : 400vh = 4 écrans de scroll. */
    .film{position:relative;height:400vh}
    .scene{position:sticky;top:0;height:100vh;height:100svh;overflow:hidden}
    .scene video, .scene img{position:absolute;inset:0;width:100%;height:100%;object-fit:cover}

    .titre{position:absolute;left:24px;right:24px;bottom:48px;opacity:0;transform:translateY(20px);transition:.6s}
    .titre.visible{opacity:1;transform:none}
    .titre h1, .titre h2{font-size:clamp(40px, 8vw, 96px);line-height:1;text-transform:uppercase}
    .suite{padding:80px 24px;max-width:720px;margin:0 auto;font-size:18px;line-height:1.6}
  </style>
</head>
<body>
  <div class="film" id="film">
    <div class="scene">
      
      <video id="video" muted playsinline preload="auto"></video>
      <!-- Chaque titre apparaît entre deux moments du voyage (0 = début, 1 = fin). -->
      <div class="titre" data-de="0" data-a=".25"><h1>Bienvenue</h1></div>
      <div class="titre" data-de=".3" data-a=".6"><h2>Entrez</h2></div>
      <div class="titre" data-de=".65" data-a="1.01"><h2>Installez-vous</h2></div>
    </div>
  </div>
  <section class="suite"><p>La suite de ton site : carte, horaires, contact...</p></section>

  <script>
    (function(){
      var
      film=document.getElementById('film'),video=document.getElementById('video'),affiche=document.getElementById('affiche');
      var titres=[];
      // Téléphone en portrait : on charge la version verticale.
      var mobile=innerWidth/innerHeight<.8;
      if(mobile)affiche.src='film-poster-mobile.jpg';
      video.poster=affiche.src;video.src=mobile?'film-mobile.mp4':'film-web.mp4';
      var cible=0,actuel=0,pret=false;
      function avancement(){var h=film.offsetHeight-innerHeight;return Math.min(1,Math.max(0,-
      film.getBoundingClientRect().top/h))}
      function maj(){
        var p=avancement();
        titres.forEach(function(t){t.classList.toggle('visible',p>=t.dataset.de&&p<+t.dataset.a)});
        if(pret)cible=p*(video.duration-.05);
      }
      // La vidéo rattrape le scroll en douceur : c'est ce qui donne l'effet « caméra ».
      function boucle(){
        if(pret&&Math.abs(cible-actuel)>.008){actuel+=(cible-actuel)*.2;video.currentTime=actuel}
        requestAnimationFrame(boucle);
      }
      video.addEventListener('loadedmetadata',function(){pret=true;maj()});
      // iPhone : une lecture très brève au premier toucher débloque le positionnement dans la vidéo.
      addEventListener('touchstart',function deb(){video.play().then(function(){video.pause()}).catch(function(){});removeEventListener('touchstart',deb)},{passive:true});
      addEventListener('scroll',maj,{passive:true});addEventListener('resize',maj);
      requestAnimationFrame(boucle);maj();
    })();
  </script>
</body>
</html>

```

YOUR TURN TO SHOOT

YOUR FIRST FILM STARTS HERE

Start small: five shots, a place you know, one weekend.
Your second site will take half the time, and your third will
be the one you show everywhere.

A question, a blocker, a site to show me? Write to me:
cadi.contactpro@gmail.com · yaniscadi.fr/liens