Gesleap

User Manual

Webcam gesture control for Linux and Windows · Edition covered: full (licensed) · Revised October 2026

At a glance

- **Start it:** run `telemouse_overlay` inside the Gesleap folder.
- **Licence:** paste the key from your purchase email on first run.
- **Use it:** move your hand to move the pointer; the shapes in Section 4 are the clicks and scrolls.
- **Change something:** hold four fingers up for two seconds to open settings on screen, or run `telemouse_settings`.
- **Stop it: index and little finger up** (see the red box on page 2).

> **Wayland users: read this before anything else**
>
> On Wayland the app cannot move the pointer into native Wayland windows until it has permission to use `/dev/uinput`. Without it the overlay appears and reacts to your hand, but clicks only land in XWayland windows – so it looks broken in a terminal but works in a browser, or the reverse. Fix it once in Section 3; it takes about a minute and a log-out.

Contents

1 **Running Gesleap** 2

2 **Activating your licence** 3

3 **Wayland: making clicks reach every window** 3

4 **Gestures** 5

5 **Settings mode (on screen)** 5

6 **The settings editor** 6

7 **Settings reference** 8

8 **Troubleshooting** 9

9 **Where things are** 9

1 Running Gesleap

Start it

The download is a folder named `Gesleap`. Keep the folder together: the executable needs the model, the icons, the Qt libraries and the Python helper that sit beside it. Moving the executable out on its own will not work.

> **How to start**
>
> **Linux** – in a terminal, from inside the folder:
>
> ```
> ./telemouse_overlay
> ```
>
> or double-click `telemouse_overlay` in a file manager.
> **Windows** – double-click `telemouse_overlay.exe`.

The first run asks for a licence key. After that it goes straight to the overlay. There is no main window to manage: the app draws a transparent overlay over the whole screen and reads the camera.

Command-line options

Most people never need these, but they are the quickest way to diagnose a problem.

Option	Effect
`--license-status`	Print the licence state and where it is stored, then exit. Works without a display, so it is safe over SSH.
`--deactivate`	Release this computer's licence slot so the same key can be used on another machine.
`--version`	Print which edition this binary is.
`--settings`	Start directly in settings mode, skipping the four-finger gesture.
`--config PATH`	Use a different `config.ini`.
`--camera N`	Use camera index `N` (default `0`); try `1` if you have more than one camera.
`--video PATH`	Read a video file instead of the camera. Useful for repeatable testing.

Stopping it

> **To stop Gesleap: show your index finger and your little finger**
>
> **Index and little finger up**, the other two folded – the shape people make for "rock on". The thumb may be either way. That is the `exit` binding, and it closes the app cleanly.
>
> Your hand must be in *action mode*. If the settings panel is on screen, gestures are only being previewed, so exit will not fire: do the four-finger gesture to come back to action mode first.
>
> If the camera cannot see you, start it from a terminal and press **Ctrl+C** there. The window has no close button by design, since a title bar over other people's applications would be in the way.

The exit shape is a preset. If you remap it (Section 6), that binding is what stops the app – so if you change it, remember what you changed it to before you need it.

> **If nothing responds**
>
> Closing the camera preview or covering the lens does not stop the app; it keeps running and waiting. Use the exit gesture, **Ctrl+C** in its terminal, or the last resort on Linux:
>
> ```
> pkill -f telemouse_overlay
> ```

2 Activating your licence

On first run Gesleap shows an activation window before anything else happens.

1. Copy the key from your purchase email. It looks like `XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX-V3`.
2. Paste it into the field – the **Paste** button reads your clipboard if you would rather not use the keyboard shortcut.
3. Press **Activate**. The window closes and the app starts.

The key is tied to one computer. It is checked over the network, so the first run needs an internet connection; later runs use the stored result.

Moving to another computer

Release the old machine *first*, from the old machine:

```
./telemouse_overlay --deactivate
```

You can also press the deactivate control in the licence panel while in settings mode. Then activate on the new computer with the same key.

When activation fails

Run `./telemouse_overlay --license-status`. It prints the state file location, whether a key is stored, the machine fingerprint and the last verification result – which is the first thing support will ask for.

Message	What it means
No licence stored	First run, or the state file was moved or deleted. Activate again.
Already activated elsewhere	The key is in use on another machine. Deactivate there first.
Could not reach the server	No network, a proxy, or the clock is badly wrong. Check the connection and the system time, then retry.

The licence state lives in the per-user application data folder (`~/.local/share/telemouse_overlay` on Linux). Deleting that folder loses the stored key; it does not free the slot on the server.

3 Wayland: making clicks reach every window

On Wayland, an application may not inject input into other applications. The X11 test-extension path that older tools use only reaches XWayland, which is the compatibility layer, not native Wayland windows. The result is selective: the pointer moves everywhere, but clicks land in some windows and are ignored in others.

Gesleap handles this by creating a real kernel input device through `/dev/uinput`, which the compositor accepts as ordinary input. That device must be writable by your user.

The one-command fix (Linux bundle)

```
sudo ./setup_uinput.sh
```

Then **log out and back in** – group membership is read at login, so without that step the new rule has no effect for your session.

What the script does, if you prefer to do it by hand

1. Create the file `/etc/udev/rules.d/99-telemouse-uinput.rules` containing exactly this one line (the file number matters – see the warning below):
   ```
   KERNEL=="uinput", MODE="0660", GROUP="plugdev", OPTIONS+="static_node=uinput"
   ```
   Any root editor will do: `sudo nano /etc/udev/rules.d/99-telemouse-uinput.rules`.
2. Load the module now and at boot:
   ```
   echo uinput | sudo tee /etc/modules-load.d/uinput.conf
   sudo modprobe uinput
   ```
3. Apply the rule:
   ```
   sudo udevadm control --reload-rules
   sudo udevadm trigger --subsystem-match=misc --sysname-match=uinput
   ```
4. Check the group:
   ```
   ls -l /dev/uinput
   ```

You want `crw-rw---- root plugdev`. If it reads `input` instead, another rule is winning.

5. Make sure you are in the group, then log out and back in:
   ```
   sudo usermod -aG plugdev $USER
   ```

> **The udev quirk: rule order decides, and the distro gets there first**
>
> Ubuntu and Debian ship their own rule, `/usr/lib/udev/rules.d/80-uinput.rules`, which sets the device group to `input`. udev applies rules in numeric order and **the later one wins**, so a Gesleap rule numbered below 80 is silently overridden. Nothing reports an error: the file appears correct, the reload succeeds, and `/dev/uinput` still belongs to `input` – which is why the app keeps falling back.
> That is why the rule is numbered `99-`. If the group is still wrong afterwards, find what is outranking it:
> ```
> grep -rn uinput /usr/lib/udev/rules.d/ /etc/udev/rules.d/
> ```
> and give your rule a higher number than anything it prints. Older versions of the setup script used `70-telemouse-uinput.rules`; it is worth deleting that file if it is still present, since it never worked.

If you skip this

The app still runs. It falls back to X11 injection and shows a startup diagnostic banner saying so, and clicks only reach XWayland windows. That banner is the thing to look for at the top of the screen on the first run.

> **This is a real permission, so treat it as one**
>
> Access to `/dev/uinput` lets a program synthesise keyboard and mouse input system-wide. Grant it to a trusted group only, and do not add users who should not have it. Gesleap uses it only to move the pointer and click.

X11 sessions do not need any of this, and Windows has no equivalent step.

4 Gestures

How a gesture is written

A gesture is five characters, one per finger, in this order:

thumb	index	middle	ring	little
1st	2nd	3rd	4th	5th

Each character is `1` (extended), `0` (folded) or `X` (either way). So `X1000` means "index up, thumb either way, the other three folded". Two-handed gestures join two codes with `+`, in screen order – left hand first:

`X1000+X1000` = index up on both hands

`X` exists so ordinary shapes are not rejected over the thumb: almost nobody holds their thumb identically twice.

The preset gestures

These ship with the app and are what you get on a fresh install.

Shape	Code	Does
Fist (nothing up)	`00000`	Nothing. A safe pose to rest your hand in.
Index up	`X1000`	**Left click**
Index + middle	`X1100`	Double click
Index + middle + ring	`X1110`	Right click
Middle + ring + little	`X0111`	Scroll up
Little only	`X0001`	Scroll down
Ring + little	`X0011`	Scroll left
Middle + ring	`X0110`	Scroll right
Open hand, all five	`11111`	Start dragging (holds the button down)
Four fingers, no thumb	`01111`	Drop – releases a drag
Index up, thumb folded	`X0100`	Select: holds the button, releasing ends it
Index + little	`X1001`	**Exit Gesleap**
Index up, both hands	`X1000+X1000`	**Ctrl+C**
Index + middle, both hands	`X1100+X1100`	**Ctrl+V**
Index + middle + ring / index	`X1110+X1000`	**Alt+F4**
Index + middle + ring, both	`X1110+X1110`	Idle: pointer pauses for a moment

Any value that is not one of the actions above is sent as a keystroke, so `Ctr+p` or `alt+Tab` work as they read. A keystroke the app cannot parse does nothing at all – it is never handed to a shell.

> **The four-finger shape does two jobs**
>
> Four fingers up with the thumb folded is the `drop` action *and* the settings-mode toggle. A quick flash of the shape drops; holding it for two seconds opens settings. If you are not dragging anything, drop does nothing, so the overlap is harmless in practice.

Making gestures easier or harder

Three settings in `config.ini` (or the settings editor) control how forgiving the engine is:

- **Frames per gesture** (default 8) – how long a shape must be held before it counts. Lower fires faster and misfires more.
- **Gesture cooldown** (default 425 ms) – how long the engine waits after firing before accepting the next gesture.
- **Pointer smoothing** (default 5) – how many recent hand positions the pointer averages. Higher is steadier and lags more.

5 Settings mode (on screen)

The overlay has its own settings panel, controlled by hand, so you do not have to leave what you are doing.

Opening and closing it

1. Hold four fingers up – index, middle, ring and little, thumb folded – for two seconds. A ring fills while you hold.

2. A **5, 4, 3, 2, 1** countdown appears. Keep the shape until it finishes; drop it and the countdown cancels.
3. The settings panel opens.

To leave, make the same four-finger shape again and wait out the countdown. You are back in action mode.

Inside the panel the pointer becomes a normal mouse pointer, so you can also click the controls directly.

What is in the panel

- **Sliders** – gesture frames, cooldown, detection, tracking, icon cooldown, smoothing, max health, stillness, and mouse-pad size. Values change live, so you can feel the difference immediately.
- **Action tiles** – pick one of the eight main gestures and cycle the action it performs.
- **Licence panel** – shows the current licence state, with **Deactivate** for moving to another computer.

Changes are written back to `config.ini` when you return to action mode. Nothing is written while you are still in the panel, so abandoning it leaves the file as it was.

> **Gestures still work in settings mode – but only as previews**
>
> Actions are not performed while the panel is open: raising the exit shape previews "exit" instead of quitting. This is deliberate, so that pointing at a control cannot also click something in the application behind it. To actually quit, return to action mode first.

6 The settings editor

`telemouse_settings` is a small standalone program that edits `config.ini` with ordinary mouse and keyboard. It needs no camera and does not start the gesture engine, so it works even when the app itself will not start.

> **Opening it**
>
> **Linux** – from the Gesleap folder:
> `./telemouse_settings`
> **Windows** – double-click `telemouse_settings.exe`.
> Give it a path to edit a different file: `./telemouse_settings /path/to/config.ini`

- **Nothing is written until you press OK. Cancel** leaves the file byte-for-byte as it was.
- **Comments survive.** The editor rewrites only the values it knows about; comments, section order and unfamiliar keys are kept.
- **It refuses duplicates.** A gesture shape can only map to one action, so a second row with the same shape is rejected rather than silently shadowing the first.
- **Reset to factory** restores every value and the full gesture list.

Gesture rows

Each row is a set of dots (click to cycle folded / extended / either; right-click to add or remove the second hand) and an action list. The list holds:

- the named actions – left click, scroll, drag, exit and so on;
- **Other key combination...** – type a keystroke such as `Ctr+p`;
- **Launch a program...** – opens a file picker and writes the quoting for you.

Mapping a gesture to a program

A binding whose value begins with a double quote is a program to run:

```
X1000="/usr/bin/blender"
X1111="C:/Program Files/Blender Foundation/Blender/blender.exe"
```

The quotes are required. Without them the value is read as a keystroke, so a path can never be mistaken for a key combination. Arguments go *after* the closing quote, exactly as a shell expects:

```
X1000="/usr/bin/code" --new-window
```

Everything inside the quotes is one path, so spaces in a path are safe. The program is started directly with no shell

in between: a path that does not exist therefore does nothing rather than running something unintended.

Mouse pad size and frame padding

These two describe the same thing from different directions, and the file now treats **frame padding as the authoritative one**. If `config.ini` sets it, that value is used and the mouse-pad size is ignored; if only the mouse-pad size is present, padding is derived from it. The editor shows them in step, so the number you type is the number the engine uses.

Worked example: open Blender with a gesture

Say you want three fingers up to start Blender, without editing any files.

1. Open the editor and find a row you do not use – **Idle**, say – or press **+ Add gesture** for a new one.
2. Click the row's dots until the shape reads **index, middle and ring extended** (code `X1110`).
3. Open that row's action list and choose **Launch a program...**. A file dialog opens.
4. Pick the Blender executable (`/usr/bin/blender`, or `blender.exe` on Windows). The editor writes the quotes for you.
5. Press **OK**, then restart Gesleap so it reads the changed file.

The row now reads **Run blender**, and the gesture opens it. To add arguments – opening a file, for instance – edit `config.ini` by hand and put them after the closing quote.

> **A program that fails to start says nothing**
>
> There is no shell involved, so a wrong path simply does nothing. If a gesture that launches a program appears to be ignored, check the path first: run it once in a terminal to confirm it starts. Spaces in the path are fine as long as the quotes are there.

7 Settings reference

Defaults are the values in the shipped `config.ini`. Editing by hand is fine; the editor simply keeps you inside the ranges below.

`[tracker]`

Key	Default	Range	Effect
`min_detection_confidence`	0.35	0–1	How sure the detector must be that a hand is in frame. Lower finds hands in poor light but invents more.
`min_tracking_confidence`	0.425	0–1	How sure it must be to keep following a hand it has already found. Lower survives partial occlusion, jitters more.
`cooldown`	425 ms	100–1500	Pause after a gesture fires before the next is accepted.
`frames_per_gesture`	8	1–30	How long a shape must be held to count. Lower fires sooner, is less stable in motion.
`icon_cooldown`	2	0–30	Frames an action icon stays hidden after being shown.
`number_of_centroids`	5	0–20	Pointer smoothing. Higher is steadier and lags behind.
`max_health`	13	10–20	Frames without a hand tolerated before the tracker resets.
`select_downtime_ms`	1000	500–3000	How long other gestures are ignored after a select ends.
`script_downtime`	3000	0–10000	Same, after launching a program.
`timeout_idle`	1 s	0.5–3	How long idle mode lasts. Decimals are accepted.
`is_still_sensitivity`	1.2	0.1–5	How much movement still counts as holding still.

`[mouse handler]`

Key	Default	Range	Effect
`frame_padding`	0.10	0–0.49	Inset from the camera frame that maps to the screen edge. Small values reach the edge with less hand travel. Authoritative when present.
`roi`	0.65	0–1	Fraction of the frame used as the mouse pad. Only used when `frame_padding` is absent.
`scroll_step`	60 px	1–500	Distance for one vertical scroll.
`scroll_step_horizontal`	60 px	1–500	Same, horizontally. On Windows this is in wheel units, where 120 is one notch.
`show_icons`	true	–	Draw the action icons and progress rings.

`[gesture extractor]`

`low_contrast_method=hist` and `do_polygon_checks=true` choose how the hand silhouette is separated from the background. The defaults suit a normal webcam; change them only if you are experimenting.

Editing by hand

- The file is plain INI. `#` starts a comment.
- Values are read at startup, so restart the app after hand-editing.
- If a gesture shape appears twice, the last one wins – which is why the editor refuses duplicates.

8 Troubleshooting

Symptom	Likely cause and fix
Overlay appears, pointer moves, clicks land in some windows only	Wayland without `/dev/uinput` access. Do Section 3, then log out and back in.
`/dev/uinput` is group `input`, not `plugdev`	An udev rule numbered higher than yours is winning. See the warning in Section 3.
Nothing happens at all on start	Check the licence: `--license-status`. A stored key that cannot be verified stops the app before it touches the input devices.
"Could not connect to display"	Running over SSH or with no desktop session. Use the console options only, or run it from the machine's own desktop.
Gestures fire when you do not want them	Raise **Gesture cooldown** and **Frames per gesture**.
Gestures take too long to register	Lower **Frames per gesture** a little, and keep your hand inside the camera's view for the hold.
Pointer is jittery	Increase **Pointer smoothing**; improve lighting; avoid a busy background.
Pointer needs lots of hand travel	Increase **Mouse pad size**, or lower **Frame padding**.
Camera not found	Another program has it, or it is not index 0. Close the other program, or try `--camera 1`.
"Launch a program" missing from the action list	The editor you are running is older than the feature. Reinstall from the current download.
Editing `config.ini` changes nothing	You are editing a different copy. The packaged app reads the one beside the executable; restart after saving.
Free edition stops after a minute	By design: a free run lasts 60 seconds, then shows "End of trial time." and closes.

9 Where things are

`telemouse_overlay`	Launcher (Linux). Sets up bundled libraries and starts the real binary.
`telemouse_overlay.exe`	The app on Windows.
`config.ini`	Gestures and tuning. Read at startup.
`telemouse_settings`	The editor described in Section 6.
`hand_landmarker.task`	Hand detection model.
`icons/`, `clfs/`, `python/`	Action icons, classifiers and the bundled Python helper.
`setup_uinput.sh`	The Wayland permission fix (Section 3).
`README.txt`	The short version of this manual.

The two things worth remembering

1. Four fingers up for two seconds opens settings; the same shape closes it again.
2. Index and little finger up quits – but only from action mode.

Gesleap. Third-party components and their licences are listed in `THIRD_PARTY_NOTICES.md` in the source tree.