A-Group-Of-Seven-Painter

Yahya Nazer

2026.06.18

Table of contents

Group of Seven Painter

A desktop tool that turns a photograph into a **Group of Seven / Tom Thomson style oil painting** — bold limited-palette color, thick palette-knife brush strokes, impasto relief, and optional dark outlines.

It runs as a simple windowed application (Tkinter GUI). You pick an image, choose a style or set the controls yourself, click **Generate**, and it writes finished `.jpg` paintings to disk.

What it does

The program rebuilds your photo as a painting in a few stages:

1. **Color** — boosts contrast and saturation for the vivid Group of Seven palette.
2. **Simplify** — smooths fine detail and reduces the image to a small *adaptive* palette (real colors picked from your photo, so hues stay true).
3. **Brushwork** — lays down short, thick, hard-edged **palette-knife slabs** of solid color that follow the shapes in the image, then adds **impasto** lighting so the paint reads as raised ridges.
4. **Outlines** — overlays bold dark edges along the strongest contours.
5. **Finish** — a gentle final sharpen.

Setting the **Stroke** control to `0` skips the brushwork and gives a clean flat "poster" version instead.

Requirements

- **Python 3.9+**
- **Pillow** and **NumPy** (the only two dependencies — no SciPy needed)

Install the libraries once:

```
pip install pillow numpy
```

Works on **Windows** and **macOS**. Tkinter ships with the standard Python installer on both; if you are on Linux and Tkinter is missing, install `python3-tk`.

Running it

From a terminal:

```
python Group-Of-Seven-Painter_V01.py
```

Or open the file in VS Code and run it with Code Runner / the Run button. A window titled **"Group of Seven Painter"** opens.

Using the window

The window is laid out top to bottom:

1. **Source image** — click **Browse…** and pick a photo (`.jpg .jpeg .png .webp .bmp .tif .tiff`).
2. **Copyright (optional)** — type a line such as `yahya Nazer @ 2026`. If filled in, it's stamped in italic white (with a soft dark outline for legibility) at the lower-right of every saved image. Leave it blank for no stamp.
3. **Mode** — choose **Set manually (ranges)** or **Read from CSV (recipes)** (see below).
4. **Style preset** — pick a named look from the drop-down to fill every control at once, then fine-tune.
5. **Parameters** — the six controls, each with an *Initial*, *Final*, and *Increment* value.
6. **Generate / Open Output Folder / Quit** — run the job, jump to the results, or close.
7. **Status line** — shows progress and where files were written.

The Style preset drop-down

Selecting a style fills in **all six parameters** (both the *Initial* and *Final* fields) so you get one ready-to-render setting. From there you can:

- Click **Generate** straight away for that single look, or
- Widen any parameter's range (set *Final* different from *Initial*) to sweep several variations in one run.

Choosing **(custom - set manually)** leaves your current values untouched.

The six parameters

Control	Range	What it does
Brush size (B)	3–21	Thickness of the palette-knife slabs (and the smoothing kernel). Small = fine dabs; large = bold chunky slabs.
Color levels (C)	4–32	Richness of the adaptive palette. Lower = a more limited, posterized Group of Seven palette; higher = more colors retained.
Smooth passes (S)	0–4	How much fine detail is cleaned up before painting. 1–2 is usually right.
Edge strength (E)	0–10	Intensity of the bold dark outlines. `0` turns outlines off.
Saturation (V)	0.8–2.0	Color vibrancy multiplier. 1.3–1.6 gives the warm autumn punch.
Stroke / impasto (T)	0–100	The painterly control. `0` = flat poster (no strokes); 60–95 = palette-knife painting with raised paint. Higher also deepens the impasto relief.

A strong Group of Seven look: Brush 9–15, Levels 12–16, Edge 3–5, Saturation 1.3–1.6, Stroke 75–95.

Built-in style presets

Style	Brush	Levels	Smooth	Edge	Saturation	Stroke
yahya-grp7	3	18	1	2	1.20	1
Thomson autumn	11	14	1	4	1.45	85
Bold palette knife	15	12	1	3	1.50	90
Fine sketch	7	18	1	5	1.35	75
Heavy impasto	12	12	1	3	1.50	95
Soft muted	10	16	2	3	1.30	65
Vivid jack pine	11	14	1	5	1.60	85
Flat poster	9	14	1	2	1.35	0

These match the values in `Group-Of-Seven-Recipes-Sample.csv`.

Two ways to run a job

Manual (ranges)

Each parameter has an **Initial**, **Final**, and **Increment**. The tool renders every combination from initial to final in steps of the increment. If you leave *Final* equal to *Initial* for a parameter, that parameter stays fixed.

Example: Brush `9 -> 13` step `2`, everything else fixed, produces three paintings (Brush 9, 11, 13). Because combinations multiply, the program **warns you before generating more than 300 images** so a wide sweep can't run away.

CSV (recipes)

Switch to **Read from CSV (recipes)** and choose a `.csv` file. Each row is one painting with its own settings — handy for keeping a repeatable batch of looks.

Columns (any order; header capitalization and spacing don't matter):

```
Index, Description, Brush, Levels, Smooth, Edge, Saturation, Stroke
```

`Index` and `Description` are used to label the output file. The sample file `Group-Of-Seven-Recipes-Sample.csv` is ready to load.

Where the output goes

Smart location: if your source image lives anywhere inside a folder named **`a-Data`** (case-insensitive), the paintings are written to a matching **`c-Results`** folder beside it. Otherwise they go next to the source image. This follows the `a-Data` / `c-Results` project convention.

Timestamped subfolder: each run creates its own folder named after the image plus a timestamp, e.g.

`<results-root>/<image-name>-20260616-111200/`

The original first: every run saves an untouched copy of your source image as the first file (`img-001-original.jpg`) so you always have the reference sitting beside the paintings.

File names encode the settings so you can tell renders apart at a glance:

`img-002-b11-c14-s1-e4-v1.5-t85.jpg`

That reads as: sequence `002`, Brush 11, Levels 14, Smooth 1, Edge 4, Saturation 1.5, Stroke 85. CSV runs also append the recipe index and a short description slug. Numbering continues from any existing `img-###` files in the folder, so reruns never overwrite earlier work.

All images are saved as high-quality JPEGs at **300 DPI**, suitable for printing. (DPI is print-resolution metadata; the pixel dimensions match your source image, so print size = pixels / 300 inches.)

The results are **deterministic**: the random brush placement is seeded from the parameters, so the same settings always reproduce the exact same painting.

> **Note:** Painting happens on a copy scaled to a maximum of 1200 px on the long edge (to keep it fast), then the result is scaled back to the original size.

Tips for best results

- **Landscapes and foliage** shine — strong shapes and warm color are what the style is built for.
- Start from a **preset**, generate once, then nudge one control at a time.
- Want **blunter, flatter slabs**? Lower Levels and raise Brush. Want **finer, busier** texture? Raise Levels and lower Brush.
- Turn **Edge** down (or to 0) if the dark outlines feel too heavy for a soft scene.
- For a quick **non-painterly poster**, set **Stroke = 0**.

Customizing the program

A few settings live near the top of the file:

- **`STYLE_PRESETS`** — the preset table. Add or edit a line here (name plus the six values) and it appears in the drop-down automatically; no other code changes needed.
- **`PRINT_FLAG`** — set to `False` to silence the `[DEBUG]` progress messages in the console.
- **`MAX_PAINT_DIM`** — the 1200 px painting cap; raise it for more detail at the cost of speed.
- **`MAX_WARN_THRESHOLD`** — the 300-image safety warning threshold for manual sweeps.

Troubleshooting

- **"No module named PIL / numpy"** — run `pip install pillow numpy`.
- **The window doesn't open on Linux** — install Tkinter (`sudo apt install python3-tk`).
- **Output looks too smooth / not enough texture** — raise **Stroke** (try 85–95) and check it isn't set to 0.
- **Colors look off** — lower **Saturation** toward 1.2–1.3, or raise **Levels** to keep more of the original palette.

- **Can’t find the results** — click **Open Output Folder**, or read the path shown in the status line after a run.