MeDFLIstvan Editor v2.64 — Brawscope

1. What Is MeDFLIstvan? Why Is Its Codename Brawscope?

The **MeDFLIstvan Editor** is a quality-focused image editor specializing in the Commodore Plus/4 **DFLI (Double Line FLI)** graphics format. The program allows users to create DFLI images, correct existing images at the pixel and color-channel levels, optimize color registers, and convert external images to DFLI.

The **codename of v2.64 is Brawscope**, formed from the names of the three development areas: **BR**ush, dr**AW**, and kaleido**SCOPE**. This release brings particularly significant enhancements to these editing capabilities. Another major innovation is **Smart Preserve**, a hardware-aware color-handling mode.

The program is not intended exclusively for demoscene developers familiar with hardware registers. Revised terminology and the separation of paint colors are designed to let users think first in terms of **visible colors and drawing operations**. Precise hardware-level control nevertheless remains available.

What does "quality-first" mean? The visible appearance of an image and the hardware limitations of the Plus/4 are given particular importance. The goal is to preserve the best possible quality and editability, **not** to promise guaranteed lossless behavior for every operation.

2. Installation and First Launch

Recommended screen resolution:

- **Full HD (1920×1080 pixels) or higher.**
  Minimum acceptable screen resolution: 1536×864, with some compromises.

System requirements:

- **Windows:** Windows 10 (x64) or Windows 11 (x64) recommended.
- **Linux:** Tested on Linux Mint 22 (amd64).
- **macOS:** Tested on Apple Silicon.

The planned/distributed platform packages **include a bundled Java runtime**, so a separate Java installation is not required when using these packages.

On launch, the main window displays the canvas, color palette, Pointer information area, Tools, I/O, and PaintChannel controls. The program also supports separate Light/Dark appearance settings. Because of the user interface layout, smaller screens may require scrolling to reach certain panels (*1536×864 or lower*).

3. How Plus/4 FLI–DFLI Works

What Is FLI, and Where Does the Name DFLI Come From?

In the classic Plus/4 graphics modes, the **TED** video controller reads screen color information according to a scheme tied to character cells. A standard multicolor bitmap character cell is relatively large, so changing a color within the cell can affect several pixels. The collective term **FLI (Flexible Line Interlace/Interleave/Interpretation)** covers techniques that alter the timing of color-information access. The Plus/4 World encyclopedia specifically notes that **TED requires two raster lines for a complete color-fetch cycle**. Accordingly, properly implemented Double Line

FLI changes the relevant color-map data every two lines. This is where the name **DFLI (Double Line FLI)** comes from. (Source: Plus/4 World — FLI.)

In more technical terms: DFLI is an extended MC (Multicolor) format that can produce approximately four times the effective local color flexibility: it uses four color RAM areas for Color1 and Color2 instead of one, while the Background (FF15) and PlusColor (FF16) color registers can also be changed on every raster line rather than having a single value each as in ordinary MC mode.

Practical significance: Local colors can change more frequently than in an ordinary character cell. This makes more detailed portraits, shadows, highlights, and complex transitions possible. The trade-off is that the drawing program must account for the fact that several pixels may reference the same hardware color source during every brush stroke.

MeDFLIstvan uses an editing model of **160×200 logical pixels**. Plus/4 multicolor pixels are stretched horizontally, so the editor displays them at a **2:1 aspect ratio**, corresponding to **320×200** at the default scale. A unit of 4×2 logical pixels therefore corresponds visually to 8×2 display pixels.

Pixel Code and Actual Color Are Two Different Pieces of Data

Each logical pixel stores a **2-bit code**. Instead of directly storing one of 128 arbitrary colors, it selects one of **four channels**. The actual Plus/4 palette color for that channel is supplied by a separate data table.

Code	Label in the program	Technical color source	Shared across
0	**Background FF15**	Register: FF15	One entire raster line
1	**Color1**	AUX1 (one FF14 color slot)	One 4×2 DFLI block
2	**Color2**	AUX0 (the other FF14 color slot)	One 4×2 DFLI block
3	**PlusColor FF16**	Register: FF16	One entire raster line

Simple example: Several pixels in a light-blue horizon use code 3, the PlusColor channel. If you set that line's PlusColor FF16 value to orange, every code-3 pixel on the same line may turn orange—not just the pixel you clicked. Changing a Color1 or Color2 block color can affect pixels using the same code within that 4×2 block. This is called a **color clash**.

Why is this difficult for beginners? In a modern RGB editor, a pixel usually has its own color. In a DFLI editor, the visible pixel color and the hardware channel carrying it must be handled separately. Brawscope aims to manage as much of this complexity as possible through its interface and Smart Preserve.

How Does This Relate to File Formats?

The Plus/4 World **Image File Formats** article describes the standard bitmap, Botticelli, Multi Botticelli, and MCFLI file-format families, including the role of color maps. This is **background information**, not proof that Brawscope can directly import every possible layout of those files. The specific supported input formats are documented in Chapter 18. (Source: Plus/4 World — Image File Formats.)

Recommended reading: Plus/4 World — FLI and Plus/4 World — Image File Formats.

4. Start Screen and Workspace

Practical Guide: Understanding Navigation—Top Bar, Tools, I/O, and PaintChannel

Brawscope does not follow the traditional, deeply nested `File/Edit/View/Tools` menu structure. The large buttons in the **Tools** section on the right generally open further selection dialogs; **I/O** is dedicated to file operations; and the top bar provides direct access to frequently used tools. **Pressing a button often does not change the image immediately:** sometimes it changes a mode, sometimes it opens a settings dialog, and some operations are executed only after a subsequent mouse click.

Before choosing a tool, establish:

(a.) whether a Selection is already active;

(b.) which PaintChannel is active;

(c.) whether Smart Preserve or Origi Hardware is selected;

(d.) whether the intended operation affects the whole image or only the selected area.

The presence of an option in the menu does not mean that every editing mode supports every other mode.

On first launch, you will see the **main editor window**. The following description follows the default desktop layout.

Where to look	Visible element	What it does and when to use it
Top left, below the title bar	**Top menu bar: File, Kaleidoscope, Blueprint, Redraw Shape, Set**	Direct access points. **Redraw Shape is covered in full in Chapter 7 (Draw)**; it does not have a separate section under Kaleidoscope simply because it appears in the top bar.
Center, large workspace	**Canvas / DFLI image**	Edit the 160×200 logical pixels at a 2:1 display aspect ratio. Left-click generally paints/draws; right-click depends on the current mode and tool.
Upper-right column	**Plus/4 Color Palette**	Selects a paint color; does not, by itself, rewrite color data in an existing image.
Below the palette	**Pointer**	Displays the logical coordinates, visible color, color code, and PaintChannel under the mouse. These need not match the paint color selected for the next brush stroke.
Middle right	**Tools: Undo, Edit, DFLI Color Map, Brush, Color Filter, Bucket Fill, Color Presets, Magic Raster Tool, Fill with Color or Pattern, Draw**	Opens most editing selection dialogs. Edit also contains other functions and Selection.
Lower-right column	**I/O: Import PRG, Export PRG+ASM+PNG, Convert Image to DFLI, Project I/O**	Import existing Plus/4 images, convert external images, save projects, and export results.
Below I/O	**PaintChannel**	Selects the paint target (**Background FF15 / Color1 / Color2 / PlusColor FF16**) and the **Smart Preserve / Origi Hardware** handling policy.
Lower window area	**Zoom % slider, Live Kaleidoscope workspace panel (when active), Status bar**	Zoom, temporary symmetry controls, and feedback about the current operation.

The **File** and **Redraw Shape** features in the top menu bar are explained in the corresponding file-handling and Draw chapters to avoid repeating the same information in two places.

4.1 Workflows Worth Starting With

Suggested workflow:

1. Open a project, import a PRG, convert a source image, or start with an empty image.
2. Check the **PaintChannel** handling mode: Smart Preserve or Origi Hardware.
3. Select a paint color and an appropriate brush, drawing tool, or selection method.
4. Save the project before complex edits; also create an **F9 Quick Snapshot PNG** if needed.
5. Use grids, the heatmap, Blueprint, and Pointer information to inspect the image.
6. Before exporting, check the visible colors and hardware color clashes.

A) New image, manual pixel art: Select a PaintChannel and paint color; create a silhouette with the Solid brush; add straight edges using Line/Box; inspect the line and block palette in DFLI Color Map. Merely selecting a paint color does not modify the image. Whether reusing a channel causes a color clash becomes apparent when you actually paint.

B) Photograph to DFLI: Load an image into the Converter; crop and resize it; preprocess brightness and color; try Fast quality first; fine-tune quality and color character; run Convert; then correct details using brushes or DFLI Color Map. The Converter preview is not always identical to the hardware-resolved DFLI result: inspect the final image on the editor canvas as well.

C) Repairing an existing demo graphic: Import the PRG; verify that the image and color data were placed as expected; use Clean Hidden AUX or Smart AUX Dither Prep if necessary; retouch carefully in Smart Preserve mode; export a compatible PRG; and save a separate native project. PRG and `.mdfli` files are not substitutes for one another.

4.2 The Three Most Important User States

Paint color = the palette value selected for the next operation.

Visible image color = the color calculated for a canvas pixel from its channel code and the local registers at that position.

PaintChannel = the color carrier to which the operation is linked.

The Pointer panel helps distinguish these three concepts. If an unexpected color changes after painting, first inspect the actual channel value shown by Pointer, then check the Smart Preserve/Origi selection.

5. Set — Display Colors for the Editor's Visual Guides

The **Set** dialog configures the **colors and transparency of visual editing guides** displayed in Brawscope. You see these guide colors in the Windows/Linux/macOS interface; they are **not painted into the Commodore Plus/4 DFLI image**. Using Set does not change the 2-bit bitmap pixel codes or the image's Background FF15, Color1, Color2, and PlusColor FF16 color data.

How to open it: Choose **Set** on the main window's top bar. **Edit** → **Program Settings** → **Set** opens the same dialog (its location in Edit is described in Chapter 12); the complete usage instructions are provided here.

5.1 The Seven Buttons in Set—in Their Actual Display Order

Exact button label	What it affects	Practical recommendation
Char grid color...	Color of the 40×25 character-cell guide grid	On a light background, choose a darker but more transparent grid. Toggle the character grid with `Caps Lock`.
DFLI grid color...	Color of the 40×100 DFLI block grid	For inspecting boundaries between adjacent 4×2 blocks, choose a clearly distinguishable shade; toggle with `D`.
Pixel grid color...	Color of the 160×200 logical-pixel grid	Medium Alpha is useful for fine contours at high zoom; toggle with `G`.
Selection color...	Pulsating visual highlight color for the selection	Choose a contrasting color distinct from those in the graphic being edited.
Heatmap color...	Color of the channel-diagnostic Heatmap overlay	Helpful when the image's own colors obscure the pixels associated with the selected PaintChannel. Toggle Heatmap with `H`; see Chapter 21.
Outer area color Plus/4	Color of the area **outside** the logical image, i.e., a simulated Plus/4 border color	Choose a Plus/4 palette color that complements the image.
Close	Closes the Set dialog	Individual guide-color changes are committed with **Apply** in their respective color-picker dialogs.

5.2 RGB and Alpha—How to Adjust Them

The first five buttons listed above open similar detailed color dialogs, with **Red**, **Green**, **Blue**, and **Alpha** controls, each providing a **0–255** slider **and** a numeric field. These dialogs contain **Preview**, **Apply**, and **Cancel**. Preview uses a checkerboard background to show transparency. **Apply** saves the guide color in the interface configuration, applies it to the canvas, and closes the small dialog. **Cancel** discards any changes not yet applied in that color picker.

Meaning of Alpha: `255` = fully opaque guide color; `0` = fully transparent, making the guide almost or entirely invisible. This four-component RGBA control applies **only to the Set visual guides**; it must not be confused with the Brush Opacity setting in Chapter 6. Mouse behavior and DFLI register/image data remain unchanged.

Exercise 1—readable pixel grid: Choose a dark image, zoom to 400%, and toggle the `G` pixel grid. Open **Set** → **Pixel grid color...** and start, for example, with `R=255, G=220, B=80, Alpha=120`, then click **Apply**. If the grid still obscures fine detail, try `Alpha=70`; if it is barely visible, increase it to 180. These example values are suggested starting points, not factory presets. Examine the result on the canvas, not in a PRG/PNG export.

Exercise 2—highlighting a selection: If a selection is hard to see against a light-blue sky, try a saturated red **Selection color** with `Alpha` around `200`. If the overlay is too strong, reduce Alpha. Click **Apply** and inspect both the selection boundary and the edited pixels.

Exercise 3—Heatmap inspection: Press `H` to display the selected PaintChannel's diagnostic overlay, then use **Heatmap color...** to select a conspicuous, moderately transparent color. The Heatmap is not a new Plus/4 palette color or a filter: it helps you interpret the channel distribution within the image (see Chapter 21).

5.3 Why Is Outer Area Color Plus/4 Different?

Outer area color Plus/4 does not open four RGB/Alpha sliders. Instead, it uses a dedicated **Plus/4 palette-index picker** (`PaletteIndexChooser`). Selecting a new color stores and displays it for the area outside the image; canceling the selection retains the previous value. **The actual 160×200 image area and its DFLI color map remain unchanged.** This is a display preference, not a paint channel. It lets you preview how the finished image

might look on a Plus/4 with different border colors ($FF19). It is **only a border-color simulation**, not an exported hardware setting.

Limitations and checks: Set is not intended for changing the color of a particular raster line in the image. The visual grids have their own toggle keys; adjusting a grid's color does not turn that grid on. Contrast may vary between displays, so the suggested RGB/Alpha values should be treated as starting points.

6. Brush — Freehand Painting, Settings, and Channel Handling

The **Brush** is used for freehand painting and pixel retouching on the canvas. Once a Brush mode has been selected, the size, placement of brush stamps, paint color, and **Smart Preserve / Origi Hardware** policy jointly determine the result. Some modes rearrange existing pixel colors or channel codes: Plus/4 DFLI does not store arbitrary, partially transparent RGB pixels.

6.1 Overview of the Eight Brush Modes

Mode	When to use it	Details
Solid	Basic painting, outlines, and individual pixel corrections.	6.4
Pattern	Repeating surface patterns and textures.	6.5
Dither	Optical tone blending or patterning existing channels.	**6.6**
Recolor	Targeted replacement of a selected existing color.	6.7
Shade Recolor	Changing colors within the same hue family.	6.8
Shade Luma	Lightening or darkening using the Plus/4 palette.	6.9
Shade Blend	Tonal transitions with a dedicated strength control.	6.10
Smooth	Local tone smoothing, subject to limitations.	6.11

Bucket Fill Soft Dither (Chapter 8), **Fill with Pattern / Directional Halftone** (Chapter 9), and **Smart AUX Dither Prep** (Chapter 13) are **separate features**, not submodes of Brush Dither. Accordingly, Chapter 6 describes only the dithering performed by the Brush tool.

6.2 Shared Brush Settings and Value Ranges

Setting	Range / selection	Meaning
Brush Size	**1–64**; preset sizes: 1, 2, 4, 6, 8, 10, 12, 16, 32, 64; numeric field: 1–64	Brush stamp size; the numeric keypad `+` / `-` keys also adjust it.
Opacity	**1–100**	In supported modes, reduces the density of pixels covered by the stamp; it does not blend alpha colors.
Flow	**1–100**	Contributes to the coverage calculation; in certain modes it also changes the distance between stamps.
Spacing	**1–100**	Base spacing between consecutive brush stamps.
Show coverage preview	Toggle	Previews the targeted area; it is not graphics data included in a PRG.
Stroke edge	**Hard Pixel / Anti-Aliased**	Edge handling on supported brush/drawing paths.

Setting	Range / selection	Meaning
Strength	Generally **0–100**, separately for each mode	Mode-specific effect strength for Dither, Shade Recolor, Shade Luma, Shade Blend, and Smooth.

6.3 Opacity, Flow, Spacing, and Strength — A Unified Explanation

In the supported normal Brush path, **Opacity** and **Flow** work together: `max(1, min(100, (Opacity × Flow) / 100))`. Integer division is used, so 50 × 50 produces a **calculated threshold of 25%**, **not** a color with 25% transparency. A **4×4 ordered threshold/Bayer-like pattern** determines which pixels pass and which remain untouched. Across a complete stroke, overlapping stamps, existing pixel colors, and hardware constraints mean that the proportion of pixels actually changed need not equal the threshold percentage.

In some Brush modes, **Flow** also affects the spacing between stamps. The base step is `max(1, Brush Size × Spacing / 100)`; on supported paths, this is multiplied by `0.75 + 0.25 × max(0.25, Flow/100)`. With **Brush Size 8 / Spacing 100**, the initial approximation is about 8 logical pixels at Flow 100 and about 7 at Flow 50, before the engine's final step handling takes effect. The dedicated **Shade Blend** and **Smooth** paths do not apply the ordinary Opacity/Flow Bayer thinning and Flow spacing adjustment in the same way.

Strength is not a universal transparency setting: it controls the algorithm of the particular mode. In **Dither**, especially, Dither Strength, Opacity, and Flow *all* contribute to the effective strength; see Section 6.6. Shade Luma, Shade Recolor, Shade Blend, and Smooth each have their own Strength control for the corresponding color adjustment.

Reproducible exercise — four identical areas: save a test project, create four identical 24×24-pixel patches, and draw the same Brush Size 8 / Solid stroke on each using the same paint color and channel:

Test	Opacity	Flow	Calculated threshold	What to examine
A	100	100	100	Solid stamp; hardware color clashes are still possible.
B	50	100	50	Sparser pixels.
C	100	50	50	A single stamp may resemble B, while spacing may change over a longer stroke.
D	50	50	25	Strongly thinned target pattern.

Start with a larger brush: at **Size 1**, a pixel at a particular threshold-pattern position is either accepted or rejected, which can be misleading. Inspect the result at 100% and at higher zoom. As initial experiments, try Opacity 40–60 / Flow 100 for a fine-grained surface, or 100/100 for a firm outline. These are suggested starting points, not settings guaranteed to work for every image.

6.4–6.11 Detailed Use of the Brush Modes

6.4 Solid — Basic Painting, Outlines, and Pixel Corrections

Operation: draws pixels according to the selected PaintChannel, paint color, and active hardware policy. **Result:** clean shapes with consistent colors; pixel-accurate retouching at small sizes and rapid area painting at larger sizes. **Exercise:** begin with a 1–2-pixel outline and check whether a color clash affects the background. If the next color cannot be accommodated, switch the color carrier or refine the colors in DFLI Color Map. **Modern equivalent:** the logic of basic Pencil/Brush tools, but subject to hardware channel limitations.

Hard Pixel and Anti-Aliased in Brush: for the **Solid** brush, **Stroke edge** lets you choose between **Hard Pixel** (sharp pixel edges) and **Anti-Aliased** (AA, smoothed edges).

The basic idea behind AA:

With **Anti-Aliasing (AA)**, the edge of a diagonal or curved line is not represented solely by fully on or fully off pixels. The system may introduce **pixels of intermediate color or brightness** between the shape and its background, making the line appear smoother to the human eye. A conventional diagonal drawn in only two strongly contrasting colors often appears stair-stepped; this effect is called **aliasing**.

Anti-Aliased rendering therefore attempts to make a curve less jagged through additional tonal steps: the brightness of pixels near the edge visually bridges the line and the background.

Important: Solid Brush AA operates within the limitations of the shared Plus/4 DFLI color carriers. **Chapter 25** provides the detailed explanation applicable to both Brush and Draw. **Stroke edge does not affect every Brush mode in exactly the same way:** this section describes Solid's edge handling; other specialized brushes may use their own Strength and pattern-processing paths. After switching modes, check which settings are actually active.

6.5 Pattern — Materials, Decoration, and Repeating Structures

Operation: repeats a selected pattern across brush stamps. The implementation provides a **Pattern** selector, **Lock to canvas**, and **Invert**. A canvas-locked pattern follows the same image-coordinate system across separate strokes, particularly useful for brickwork, fabric, tiled floors, and repeating backgrounds. Invert can reverse the relationship between filled and empty pattern positions. **Test:** draw two adjacent lines with and without Lock to canvas, and inspect how their patterns meet.

6.6 Dither — Normal Morph and Channel Pair Serve Different Purposes

Dithering alternates pixels from two available tones or channels in an ordered pattern to create the *impression* of an intermediate shade when viewed from a distance. It **does not** create a new freely blendable Plus/4 hardware color. The **Dither mode** selector in Brush Settings provides two fundamentally different methods:

Brush Dither mode	What changes?	When to use it
Normal / **Normal Morph**	Rearranges **existing 2-bit pixel codes** toward the selected dither topology; on the described path it **does not write new color-register or color-map values**.	Refining an existing boundary between two tones or an existing dither structure while retaining the original color set.
Channel Pair	Assigns the pattern's **A/B positions to two separately selected PaintChannels**. Pixel channel codes alternate; the visible result depends on the colors carried by those channels locally.	Deliberately creating an optical shade using two channels.

Normal Morph also uses an improved, hardware- and coverage-aware processing path based on **8×4 logical-pixel windows**. It is not merely *a weaker version of Channel Pair*: **Normal starts with the existing color allocation**, whereas **Channel Pair patterns between two selected channels**. Normal's result depends strongly on the pixel distribution already present in the source image.

Dither settings include **pattern type**, **Lock to canvas** (fixed pattern phase), **Invert** (reverses the pattern), **Dither Strength 0–100**, and, for Channel Pair, the **A/B channel selectors**. Effective Dither Strength is calculated using integer arithmetic: first `Dither Strength × Opacity / 100`, then `× Flow / 100`, with the result constrained to 0–100. Reducing Opacity or Flow may therefore **also reduce the dithering effect**, rather than merely thinning an ordinary Solid stamp.

Practical comparison: on a saved surface containing two blue shades, try **Normal** first and observe how the existing pattern is rearranged. Then return to the same starting state and try **Channel Pair** with two channels that carry genuinely different local colors. If Color1 and Color2 contain the same color in a particular 4×2 block, alternating the channels **will not necessarily produce a visible transition**. **Edit → Smart AUX Dither Prep** (Chapter 13) addresses this situation: it is a **separate preparatory image modification**, and it can change visible pixel colors when both channels are used but contain identical colors.

Interpreting Dither Strength: in Normal Morph, 0% means preserving the existing pixel topology, while 100% means working toward the complete target pattern. In Channel Pair, the same control influences the A/B pattern ratio. Consequently, the same numeric setting in the two modes does not necessarily produce the same apparent effect strength.

Limitation: neither Normal nor Channel Pair can bypass the shared Background FF15 / PlusColor FF16 raster colors or the Color1/Color2 block colors. Examine the pattern at high zoom and the intended optical tone at normal size. Bucket **Soft Dither** (Chapter 8) thins only the **inside edge** of a filled region; **Directional Halftone** (Chapter 9) controls spatial distribution. Neither is an additional setting of this Brush algorithm.

6.7 Recolor — Targeted Replacement of an Existing Color

Recolor is not the same as painting over everything under the cursor. The brush first uses an appropriate **source color** (the interface provides a way to sample the source color), then operates according to the selected target and scope. **Exercise:** replace only a particular blue shade on a coat with red, while retaining the surrounding black outline. Recolor offers finer control over which visible pixel colors are included in the replacement. Any block- or raster-wide side effects are still determined by the selected hardware policy.

6.8 Shade Recolor — Adjusting a Hue Family

The Plus/4 palette contains shades of differing brightness, allowing a detail to be adjusted without changing to an entirely different color. Shade Recolor is intended to make changes within a hue family using **Shade Recolor Strength 0–100**; the implementation also includes a separate **Target Tint** dialog. **Result:** a change in the tone of skin or fabric without entirely losing its color character. **Useful for:** portrait retouching, unifying textures, and balancing sunlit areas.

6.9 Shade Luma — Lightening or Darkening Rather Than Changing Hue

The interface provides **Lighten / Darken**, **White / Highest color luma** target modes, and **Smooth / Poster 5** tonal options, together with **Shade Luma Strength 0–100**. Additional **1/2/3** choices are available for Poster stepping. The purpose is to change brightness deliberately within the tonal steps provided by the Plus/4 palette. **Exercise:** lighten the upper curve of a metallic sphere and darken its lower shadow. With Poster 5, the adjustment may produce more pronounced, poster-like tonal steps. The exact result depends on the local palette and hardware state.

6.10 Shade Blend — Subtle Tonal Blending

The **Shade Blend Strength 0–100** control changes the intensity of the transitional effect. Try it when a direct recoloring is too strong but you want intermediate shades to appear more visually consistent. **Examples:** a face-to-neck transition, light and shadow on rocks, or the shading of rounded objects. Do not interpret it as unrestricted

interpolation between two RGB colors on a Photoshop layer: the limited Plus/4 palette still determines what is possible.

6.11 Smooth — Local Shade Smoothing

Smooth attempts to produce a more balanced distribution of neighboring shades, with a separate **Smooth Strength 0–100** control. **Strength:** it may help with small, noisy tonal jumps in an otherwise well-prepared DFLI image, such as a face or clouds. **Limitation:** it is not a general-purpose Gaussian Blur; the algorithm may protect black areas and distinct hue families. **Test:** smooth the same 4×2 block boundary at 20–30 and then at a higher strength, and assess the actual visible result at normal size as well as at high zoom.

6.12 Mouse Controls and Quick Pixel Retouching

Left mouse button: starts a normal Brush stroke. The active Brush mode, paint color, Size, and Spacing determine the stamp. **Alt + left-click:** eyedropper; samples the visible pixel color/channel. **Ctrl + left-click:** opens the 4×2 DFLI block recoloring dialog; it does not merely switch the channel of one pixel. Holding **P** locks supported brush operations vertically; holding **L** locks them horizontally. When **Recolor** is waiting for a source-color sample, the first left-click selects that color rather than painting.

Right mouse button: in supported conventional Brush modes, cycles the affected pixels' **2-bit channel codes** (`0 Background FF15 → 1 Color1 → 2 Color2 → 3 PlusColor FF16 → 0 Background FF15 again ...`). It **does not add a new Plus/4 color value**. The `cycleBrush` operation may affect the entire active brush mask: use **Size = 1** for pixel-accurate corrections. If two channels carry the same color, the pixel code can change without a visible difference. Normal right-click channel cycling is disabled in **Shade Blend** and **Smooth**. With an active Draw tool, right-click may have another function, such as completing a Polyline or canceling a Curve. Right-clicking the palette relates to the PlusColor FF16 paint color, not channel cycling on the Canvas.

Pixel-correction test: make a project copy → set Size 1 → enable the Heatmap with `H` and check the affected channel → disable the Heatmap → right-click once → inspect the new channel and visible color in the Pointer panel → use Undo if necessary. If the desired color is not available on any of the four local carriers, cycling channels alone cannot create it; use a different PaintChannel, Smart Preserve, or DFLI Color Map.

6.13 Brush or Draw? Final Comparison and Limitations

Brush is suited to organic strokes, textures, and fine local retouching; **Draw** (Chapter 7) is suited to regular straight lines, curves, shapes, and geometric copies. **Shade Blend versus Smooth test:** try each mode separately on saved copies of the same banded tonal area. Shade Blend uses its own tonal blending engine, while Smooth attempts to even out shade steps belonging to the **same hue family**. Some black areas and differing hue families are protected. Smooth is **not a general-purpose Gaussian Blur**.

7. Draw — Lines, Curves, Shapes, and Anti-Aliasing

Why Is a Separate Draw Engine Needed Alongside the Freehand Brush?

The Brush follows the mouse path; Draw tools, by contrast, generate shapes according to geometric rules. A straight line is defined by two endpoints, a triangle by its vertices, and a curve by its control points. The benefits are cleaner and more reproducible outlines, adjustable thickness, and more predictable shape geometry even when zoomed in.

Tool	What it creates	Practical example
Line	Straight line segment	Building edge, horizon, light streak
Polyline	Connected line segments	Mountain ridge, geometric path
Smooth Polyline	Smoothed, gentler polyline	Rolling hills, folds in clothing
B-Spline	Continuous curve shaped by control points	Strand of hair, leaf, decorative wave
3-Point Curve	Arc defined by three points	Smile, pipe bend
Box	Rectangle/outline and fill	Window, frame
Circle	Circle/ellipse-like geometric drawing	Sun, wheel, button
Triangle	Triangle	Mountain peak, arrow
Star	Star with 3–12 points (default: 5)	Emblem, ornament
Redraw Shape	Modified copy of the last eligible shape	Repeating lights, radial motifs

Polyline, Smooth Polyline, and B-Spline start from a **shared settings dialog**, but use different geometric processing. For many shapes, you can set **Line Thickness (1–64)**, the target channel/color, **Hard Pixel** or **Anti-Aliased** edges, and collision warnings. The **Apply** button in an options dialog generally **activates the drawing tool**; the actual drawing is made afterward by interacting with the canvas using the mouse.

Hard Pixel and Anti-Aliased in Draw: the **Edge** setting for Draw shapes selects either a sharp edge aligned with the logical pixel grid or a smoother-looking edge using intermediate tones. Anti-Aliased does not mean freely blended RGB alpha edges: the available intermediate colors are limited by the local Plus/4 color carriers. The concept is explained in Section **6.4**, while a comparison of Brush and Draw and practical AA examples appear in **Chapter 25**.

Recommended Drawing Workflow

1. Decide whether the shape should have a **sharp pixel-art** appearance or a **smoother, painterly** style.
2. Select a shape and set its thickness, color/channel, edge style, and, where applicable, fill.
3. Draw the shape on the canvas; if needed, try the same shape in a separate version.
4. In Anti-Aliased mode, check the intermediate shades and any color-clash warnings.
5. For repeated geometry, Redraw Shape / Draw Copy may be faster than drawing from scratch.

Draw: Understanding the Actual Mouse Gestures and Completion Steps

Do not confuse Apply in the options dialog with completing a drawing! In many cases, Apply merely puts the selected tool into a ready state. The actual drawing consists of subsequent clicks, mouse drags, and a completion gesture on the canvas. The program checks for an active specialized Draw tool first, preventing mouse input from accidentally falling through to ordinary Brush painting while a drawing operation is in progress.

Tool	Verified interaction pattern	Practical use
Line	Separate handlers for mouse press, drag, and release	Roofline, perspective line, lettering outline.
Box, Circle, Triangle	Press → drag → release; shape-angle/position adjustment may follow	Clean frames, circles, and geometric forms.

Tool	Verified interaction pattern	Practical use
Star	Left-button press → drag → release → **select Angle** → finalize	Star emblem, badge, decoration.
Polyline	Collects nodes through **clicks**; right-click completes and applies it after at least two points, otherwise cancels it	Mountain ridge, angular outline, polygonal decoration.
Smooth Polyline and B-Spline	Use the shared Polyline-based point-entry path, with separate geometry engines	Wavy ribbon, hair, soft neon arc.
3-Point Curve	Three left-clicks: **Start** → **End** → **Control**. The curve is completed after the third point; **right-click = cancel**	Smile, convex arc, edge of a spotlight.

The **3-Point Curve** sequence is unusual: it is not Start–Control–End but **Start–End–Control**. First specify where the curve begins and ends; use the third click to determine the direction and degree of its bend. End and Control may also use virtual drawing coordinates outside the visible screen, allowing an arc to extend beyond the image boundary. **Example:** for a high arch spanning a 160-pixel-wide image, place its endpoints near opposite sides of the image and then move the Control point above the top edge.

Completing a Polyline: left-click to set the starting point, then add at least one more node; right-click to close the chain and create the drawing from the collected points. If only one point has been placed, right-click acts as *Cancel*. This differs from the normal Brush's right-click channel cycling.

Practical comparison: draw the same object outline using Polyline, Smooth Polyline, and B-Spline. Polyline retains angular joints; Smooth Polyline softens the connections; B-Spline estimates a continuous curve from the control points. Test with Hard Pixel first, then repeat with Anti-Aliased edges. Because DFLI color carriers are shared, AA may trigger a collision warning; check whether the surrounding visible colors remain unchanged.

Hard Pixel and Anti-Aliased — Targeted Draw Test and Color Verification

Hard Pixel: the edge is drawn sharply; logical pixels either belong to the shape or do not. This may be the correct choice for retro lettering, outlines, or hardware-faithful demoscene graphics. Visible stair-stepping may also be an intentional pixel-art style.

Anti-Aliased: the geometric edge is not treated as a simple yes/no decision. The algorithm attempts to realize intermediate visual coverage or tones near the edges. The eye perceives these as smoother curves or diagonals. Typical uses include eyebrows, facial profiles, spheres, rounded logo elements, and fine decorative lines. **In DFLI, an intermediate tone can be displayed accurately only when the available hardware colors and channel assignments permit it.**

Test exercise: draw the same thin 3-Point Curve once using Hard Pixel and once using Anti-Aliased, against a background containing both light and dark shades. Compare (1) the perceived smoothness of the edge, (2) the visible colors of immediately neighboring pixels, and (3) whether Smart Preserve produces collision warnings. The AA result **is not identical for every color configuration**; this is not an unrestricted 24-bit RGB canvas.

Smart Preserve and AA: Smart Preserve attempts to protect colors outside the target area wherever possible, but AA may require additional intermediate target pixels. Therefore, claiming that “Smart Preserve + AA = guaranteed losslessness” would be incorrect. Check such images visually and with pixel-accurate comparisons.

Draw Options Dialogs — Exact Controls and Usage Order

This section describes controls actually present in the **Draw** → **tool** → **Options** dialogs. Selecting a shape from the menu first opens its options dialog. **Apply** does not mean the image is already drawn: it activates the operation to be performed on the canvas. **Cancel** exits the options dialog without starting the drawing.

Shared controls. In the examined Draw Line, Box, Circle, Triangle, and Star dialogs, **Target paint channel** offers four choices: **0 Background FF15, 1 Color1, 2 Color2, 3 PlusColor FF16**. Clicking the **Color** swatch opens the Plus/4 palette selector. The **Line thickness** slider and numeric field are linked controls with values from **1–64**, in steps of one. Under **Edge**, choose **Hard Pixel** or **Anti-Aliased**. The **Warn and mark collisions on canvas (can be aborted)** checkbox is enabled by default. This is useful when a shape could also affect existing image content through a shared DFLI color value.

Line — precise straight lines. Select the channel, color, thickness, and edge mode, then click **Apply**. Drag from the starting point to the endpoint on the canvas to draw the line. **Example settings:** a 1-pixel Hard Pixel line for a strict architectural outline; a 2–3-pixel Anti-Aliased line for experimenting with the edge of a shiny, rounded object. Because AA may require additional colors, examine the surrounding image as well.

Polyline, Smooth Polyline, and B-Spline — one dialog, three results. All three menu items open the same options dialog, but the previously selected tool activates a different geometric processing method. In addition to **Target paint channel**, **Color**, **Line thickness**, and collision warnings, further curve-related options may be available. Interpret those settings only in the context of the mode actually selected. Create the shape by placing successive points on the canvas. **Example:** with the same five points, Polyline produces sharp corners, Smooth Polyline aims for gentler connections, and B-Spline aims for a more continuous curve. The completing right-click is not the normal Brush's channel-cycling operation.

3-Point Curve — order of the three clicks. After accepting the corresponding options dialog, left-click to place **Start**, then **End**, and finally **Control**. **Example:** draw a slightly curved smile; adjust the amount of bend by moving the third point farther away or closer. Right-click cancels this operation. Hard Pixel and AA are subject to the same DFLI color constraints as with other Draw tools.

Box, Circle, Triangle — shared shape options. In addition to color, channel, thickness, and edge mode, the **Fill the entire object with the same color (whitewash)** option is disabled by default. With it off, you draw an outline; with it on, a filled shape. The source's options dialog includes specific guidance: the first click sets the top-left starting corner, holding **Shift** permits independent X/Y stretching, and angle/position adjustment follows mouse-button release. **Example:** Box + Hard Pixel with Fill disabled for a window frame; enable Fill for a solid panel. The precise mouse gesture for the subsequent move/rotation step described in the source still requires graphical verification.

Star — point count and drawing sequence. In **Star points**, select a star with **3–12 points**; the **default is 5**. This refers to the number of projecting star tips, **not** the number of clicks required on the canvas. A separate **Preview** shows the shape, **Line thickness** ranges from 1–64, and **Fill** is disabled by default. After accepting the options, **press and hold the left mouse button on the canvas, drag to set the star's size, then release**. **Shift stretching:** while setting the size, hold **Shift** to adjust the horizontal and vertical dimensions independently (**Stretch XY**); without Shift, the program uses its usual aspect-ratio handling. After release, set/select the **Angle**; the shape is only finalized after this step. **Example:** five points, 1-pixel Hard Pixel, Fill disabled for a fine emblem outline; enable Fill for a solid badge. The point count controls geometry, not the number of color shades.

What should you check after drawing? The selected color and channel do not constitute an independent RGB layer: the pixels depend on shared DFLI color slots. Check the modified shape at 100%, zoom in if necessary, enable the Heatmap, and examine original image details near collisions. Until an eligible shape has actually been completed, **Redraw Shape** is disabled in the Draw selector; it becomes available after a supported Draw shape has been successfully created.

Redraw Shape — Drawing Additional Copies of Draw Shapes

Purpose and availability. Redraw Shape belongs to the Draw tool group. It uses the most recently completed, supported geometric shape to create additional copies. It is neither a Kaleidoscope symmetry mode nor a general-purpose, unrestricted SVG object system.

Controls and ranges. The dialog lets you specify the copy's **X** and **Y** position and **Rotation** angle. The checked ranges are **X −161…321**, **Y −200…400**, and **Rotation −359°…+359°**. **New feature — Scale:** if the last shape was a **Box, Circle, Triangle, or Star**, the Redraw Shape dialog also provides a **Scale** slider and numeric field, ranging from **10–800%**, with **100%** as the default. Scaling is centered on the geometric midpoint of the original, unrotated shape's bounding box. **100%** retains its original size, **50%** creates a smaller copy, **200%** a larger one, and **800%** aims for eight times its geometric extent; portions extending beyond the canvas may be clipped. Scale is not available in the Redraw dialogs for **Line, Polyline, Smooth Polyline, B-Spline, or 3-Point Curve**. Changing Scale updates the live preview **both during ordinary Draw use and while Live Kaleidoscope / Radial is active**; it does not retroactively modify the original shape. **Draw Copy** actually redraws the shape on the DFLI canvas with the selected settings. **Close** closes the dialog but **does not undo** copies already made; use **Undo** to remove them if necessary.

Step-by-step — a wreath of stars. (1) Select Draw → Star, create a star, and complete the drawing operation. (2) Open **Redraw Shape**. (3) Choose an X/Y position and Rotation different from the original; try **Scale 75%** for the first copy and **125%** for the next. (4) Check the preview and click **Draw Copy**. (5) Change the position, angle, and Scale again, then repeat. Several stars can form a wreath, radial decoration, repeating emblem, or linked motif. The same approach works with supported triangles. When **Live Kaleidoscope / Radial** is active, Scale also resizes the geometry used to generate the symmetrical copies.

What appears, and what should you watch for? The original shape remains in place, while its copy appears at the new coordinates. A successful copy may become the source shape for a subsequent Redraw operation. The coordinate ranges allow shapes to be placed beyond the canvas, so they may disappear from the visible area partially or entirely. Draw Copy changes actual DFLI image data, meaning the shared color values and color-clash constraints still apply.

Project saving: version 7 of the `.mdfli` project format saves the Redraw Shape descriptor of the last successfully completed, supported Draw shape. After reopening the project, you can therefore create another Draw Copy from that shape. The currently open but unfinalized Redraw preview's **X/Y/Rotation/Scale** working state is not saved separately. After loading, a new Redraw operation starts with **Scale = 100%**. Scale is not a persistent property of the original shape, but a working setting for the current Redraw copy; no new project-file format is required for it.

Some parsing errors caused by invalid numbers entered into numeric fields do not always produce a clear warning. For precise results, enter valid values or use the available adjustment controls.

8. Bucket Fill — Filling a Contiguous Color Area

Bucket Fill is a separate tool in the Tools section (**Ctrl+B**). Starting from a point specified on the canvas, it finds a **four-way-connected area of the same visible Plus/4 color** and modifies it using the selected target channel and paint color. It does not fill every occurrence of that color elsewhere in the image. Bucket Fill is not the same as the selection-based **Fill with Color or Pattern** tool (Chapter 9).

8.1 The Bucket Fill Options Dialog

Open the dialog using **Tools** → **Bucket Fill** or **Ctrl+B**. **Apply** accepts the settings and activates the Bucket tool; **it does not fill the image yet**. A subsequent **left-click on the canvas** starts the fill. The **Cancel** button closes the options dialog without filling anything.

Control	Options / Default	Function
Target paint channel	**0 Background FF15; 1 Color1; 2 Color2; 3 PlusColor FF16**	The target color carrier for the fill.
Color	Plus/4 color swatch / palette selector	The requested target palette color.

Control	Options / Default	Function
Edge style	**Hard** (default) or **Soft Dither**	A solid target mask or a dithered inner edge.
Softness	**1–32 logical pixels**, default **4**	Controls the width of the Soft Dither transition; the slider is disabled in **Hard** mode.
Warn and mark collisions on canvas (can be aborted)	Enabled by default	Displays warnings about potential color clashes.
Apply / Cancel	—	Activate Bucket Fill / exit the options dialog.

Important: Color and **Target paint channel** are separate settings. The same visible color may be available on different hardware channels, and changing a shared raster-line or block color can affect other pixels. The fill follows the **Smart Preserve / Origi Hardware** color-handling policy currently selected in the main window.

8.2 Hard and Soft Dither — What Changes at the Pixel Level?

Hard: Uses the entire target mask of the connected area whose visible color matches the clicked pixel. Whether the requested color can actually be represented by the hardware depends on the selected color-handling policy and the outcome of any collision handling.

Soft Dither: Starts from the same area but selectively omits target pixels **only inside that area**. It calculates a **four-directional pixel distance** from the boundary, then uses a **4×4 Bayer threshold pattern** to gradually increase the density of filled pixels toward the interior. Pixels outside the original area are not added to the target mask. This is **not** RGB alpha transparency, Gaussian Blur, or another Brush Dither mode (Chapter 6).

Softness ranges from **1 to 32**: higher values may produce a wider inner transition. In a very small region, particularly a single-pixel spot, the Bayer threshold may cause **every target pixel to be omitted**; in that case, the image remains unchanged. Soft Dither does not create new intermediate Plus/4 hardware colors. Instead, it produces an optical edge transition by retaining or skipping individual pixels.

8.3 Selection, Smart Preserve, and Origi Hardware

When a Selection already exists, Bucket Fill checks the starting point as well: if it lies outside the selection, no fill mask is generated. The search remains **within the selection** and does not cross its boundary. For Rectangle, Freehand, and Smart Area Selection, the currently active selection mask determines the eligible area (see Chapter 12 for details). This rule concerns the geometric target area; the **side effects of shared hardware color registers** must be assessed separately.

Smart Preserve: The preservation-oriented hardware planner processes the final Hard or Soft target mask. Shared color slots may require a collision warning or an **OK / Safe / Cancel** decision. If a Smart plan is invalid, Bucket Fill does not silently fall back to an Origi Hardware fill. **Origi Hardware:** Uses the earlier, more direct Bucket/collision-handling path. Because of clashes, **Safe** may modify a smaller area than originally requested; always inspect the final result.

8.4 Step by Step — Creating a Cloud Edge

1. Save a separate `.mdfli` project copy and choose a contiguous cloud area of a single color.
2. Open **Bucket Fill** (**Ctrl+B**); select **Target paint channel** and **Color**, then choose **Hard** and click **Apply**.
3. **Left-click** inside the cloud to start the fill. Check the filled area and, if applicable, the collision warning.
4. After **Undo**, return to the same starting state and try **Soft Dither**, first with **Softness 4** and then, for a separate comparison, with **Softness 12**.
5. At normal display size, assess the optical transition; zoom in to inspect skipped and recolored pixels. Also check neighboring 4×2 blocks and affected raster lines.

Mouse handling: When the click that initiated the fill is completed, Bucket mode ends and the normal Brush becomes available again. The corrected event handling **blocks any accidental Brush drag associated with the original mouse press until that mouse button is released**. Therefore, begin a new Brush stroke with **a separate, new mouse press**. This safeguard does not alter the fill mask calculated by Bucket Fill.

8.5 Limitations and Verification

Bucket Fill follows only a **contiguous area of the same visible color**: a separate patch of that color requires another operation. **The same visible color** is not equivalent to **the same PaintChannel code**. Soft Dither thins the inner edge of the area; it does not blur the surrounding pixels outside it. A mouse click does not guarantee an exact color match throughout the finished image: **Smart Preserve / Origi Hardware**, shared DFLI color sources, and any collision decision all influence which pixels actually change. If the result is unexpected, check the Pointer panel, target channel, selection, and any Collision warning.

9. Fill with Color or Pattern — Fill Color, Pattern, Directional Halftone

Fill with Color or Pattern modifies a previously selected portion of the image. Solid-color filling, repeating patterns, and directional halftoning all start from the same Selection-based workflow, but they produce different target pixel masks. **Bucket Fill**, by contrast, searches for a contiguous color area starting from a click; see Chapter 8.

9.1 Choosing a Fill Operation and Making a Selection

Preparation: Select the area to modify using **Rectangle**, **Freehand**, or **Smart Area Selection** (see Chapter 12 for the selection methods). If there is no completed Selection when you open **Tools → Fill with Color or Pattern**, the program first directs you to create one. The Fill options dialog does not replace the selection process.

With a completed selection, the **Fill action** selector offers the following operations:

Operation	Purpose
Fill with Color	Apply a single target channel and paint color to the selected portion of the image.
Fill with Pattern	Alternate between two different target channels and their associated colors according to a pattern.
Directional Halftone	A spatially directed, variable-density version of the same two-channel patterning approach.
Cancel selection	Clear the selection and leave the Fill workflow.

Important distinction: Cancel selection clears the selection; it is not the same as the **Cancel** button in a settings dialog opened later. The Selection that exists before choosing the operation defines the initial Fill target area. However, changes to shared DFLI color registers may still affect visible colors outside that area (see **9.5**).

9.2 Fill with Color — Solid-Color Filling, Step by Step

Purpose: Change the color of a predefined area uniformly, such as a wall behind a portrait or a selected portion of a coat.

How to use it: (1) create a Selection; (2) choose **Fill with Color**; (3) under **Target paint channel**, select **Background FF15 (0)**, **Color1 (1)**, **Color2 (2)**, or **PlusColor FF16 (3)**; (4) choose a Plus/4 palette color under **Fill color**; (5) preferably leave **Warn and mark collisions** enabled; (6) click **Apply** and, if the program reports a collision, consider the available decisions; (7) inspect the target area on the canvas **as well as** neighboring affected raster lines and blocks.

What does it modify? The operation attempts to provide the requested channel and color for the selected target pixels. **Background FF15** and **PlusColor FF16** are shared by each raster line, while **Color1** and **Color2** are shared

by each 4×2 block. Whether the desired change is feasible, and whether the surrounding image can be preserved, therefore depends on the original image and the selected **Smart Preserve / Origi Hardware** policy. **Apply** can initiate an actual image modification; it does not merely select a paint color.

Test: Select part of a uniformly colored wall, choose **Color1** as the target channel and a different palette color, then click Apply. Check whether pixels outside the selected wall area but within the same 4×2 blocks retained their original visible colors.

9.3 Fill with Pattern — Two Channels, Two Colors, One Pattern

Purpose: Create texture or an optical tone by alternating two hardware color carriers in an ordered pattern. Pattern filling does not write freely blended RGB colors; each pattern position refers to an actual Plus/4 channel.

How to use it: (1) create a Selection; (2) choose **Fill with Pattern**; (3) set the **Channel A** and **Channel B** target channels and their two associated Plus/4 colors; (4) the two target channels **must be different**—the pre-Apply validation rejects identical A/B channels; (5) choose a pattern type and its available settings; (6) inspect the **Live Canvas preview**; (7) after Apply, check the actual canvas image and any collisions.

One group of pattern positions requests the **A** channel/color and the other requests **B**. If the two configured colors look identical at a particular location, alternating the channels may not produce a visible dither effect.

The six pattern types available in normal Fill Pattern mode:

Pattern name in the program	Expected visual character
Chessboard	Regular checkerboard alternation.
Scrambled Bayer	Ordered, but less straightforward repetition.
Striped	A striped surface.
Zigzag	A broken, zigzag rhythm.
Pseudo-noise ordered	A noise-like appearance with an ordered distribution.
Cross hatch	Cross-hatched texture.

Example: Select part of a wall; assign a lighter Plus/4 color to A and a darker one to B, using two different channels; try **Chessboard** first and then **Cross hatch**. Compare both results from the same starting image. The final appearance depends on the DFLI color-carrier arrangement in the target area.

9.4 Directional Halftone — Directional Distribution and Controls

Directional Halftone builds on the two-channel Fill Pattern system. Here, the A/B pattern ratio depends not only on local repetition but also on a **spatial distribution**. It can be used for a light spot, a directional tonal gradient, or a dot structure that gradually becomes less dense. This is **optical dithering**, not a true RGB glow or Gaussian Blur.

Control	Options / Range	Meaning
Preset (direction)	**Left → Right; Right → Left; Top → Bottom; Bottom → Top; Diagonal ↘; Diagonal ↙**	Initial direction of the spatial variation. The actual field label is **Preset**.
Distribution / Band Shape	**Linear; Circular; Square**	Linear, circular, or square distance distribution.
Dot Style	**Square; Cross**	Square or cross-shaped dot pattern.
Scale	**2×2; 3×3; 4×4** (default: **3×3**)	Size of the repeating dot cell. **2×2** is the new fine-grained **Micro Halftone** option; **3×3** and **4×4** retain their existing behavior.

Control	Options / Range	Meaning
Density	**0–100**	Basic density parameter of the pattern.
Center X / Center Y	**0–100%**	Position of the spatial distribution's center.
Angle	**0–359°**	Direction angle for **Linear** distribution; 0° points right, 90° points down.
Fade From / Phase Start	**0–100%**	Beginning of the transition range.
Fade To / Phase End	**0–100%**	End of the transition range.

Choosing a Scale: 2×2 (Micro Halftone) uses its own four-position Square/Cross dot ordering to create a finer, more tightly repeated texture. It is useful for small areas and subtle surface detail, but a 2×2 cell has fewer local coverage steps, so some tonal transitions may look more stepped. In particular, **Cross at 2×2 is a compact alternative pixel order, not a fully symmetrical cross shape**: a 2×2 cell has no center pixel. Choose **3×3** for a more recognizable clustered Cross structure (and the established default appearance), or **4×4** for larger dot cells and a different tonal character. The selected Scale works with **Linear, Circular, and Square** distributions; it does not alter the selected A/B colors, channels, or the DFLI hardware limits. Compare the Live Canvas preview and the applied result at normal size and at high zoom.

How the three distributions differ: Linear varies density along a specified direction and responds to **Angle**. **Circular** uses radial distance from the center, while **Square** uses a square-shaped distance measure. **The current Angle control does not rotate Circular or Square distribution.** Square is therefore not a general-purpose square grid that can be rotated to any angle.

Settings test — directional lighting: Make a Freehand selection on a dark wall; choose two different channels and colors of different brightness. Set **Distribution: Linear**, **Preset: Left → Right**, **Dot Style: Cross**, **Scale: 3×3**, and **Density: 50**. Change the Fade range while watching the Live Canvas preview. Next, keep the same settings and compare **Scale: 2×2** with **3×3** to see the finer Micro Halftone texture and its shorter local tonal progression; also try both **Square** and **Cross** at 2×2. Then compare with **Circular** distribution, placing the center near the upper-right part of the wall. **Square distribution / Scale 4×4** can produce another visual character, but Angle cannot rotate this distribution in the same way as Linear.

9.5 Smart Preserve, Origi Hardware, Collision, and Preview

Fill behavior depends on the current **Smart Preserve / Origi Hardware** hardware-handling policy. **Smart Preserve** attempts to preserve the original visible colors using exact color carriers wherever possible. However, this **does not guarantee perfect color preservation for every image state**: the desired target area and all surrounding colors may not fit within the DFLI hardware limits at the same time.

Collision warning: Turning off **Warn and mark collisions does not eliminate color clashes**. Some Smart Fill execution paths may apply changes despite unresolved clashes outside the target area when the warning is disabled. For important work, **keep the warning enabled** and inspect the affected **entire raster lines and 4×2 blocks** after every Apply. **Origi Hardware** may permit more direct modification of color carriers; possible side effects on the surrounding image are especially important in that mode.

Preview ≠ final image: The **Live Canvas preview** illustrates the *requested* target pattern. During the actual Apply operation, shared hardware registers, a possible **Collision** dialog, and the selected **OK / Safe Mode / Cancel** decision may change the outcome. After Safe Mode or partial application, inspect the image's actual visible colors; the preview does not certify the final hardware state.

9.6 Practical Exercises and Limitations

Single-color coat: Use Smart Area Selection on a contiguous section of a coat → **Fill with Color** → select Target paint channel and Fill color → click Apply with collision warnings enabled. Check whether the original outline and the visible colors of neighboring image details have been preserved.

Textured wall: Rectangle Selection → **Fill with Pattern** → different A/B channels and colors → **Chessboard** or **Cross hatch** → preview → Apply. Viewed from farther away, two tones may create the impression of a new, optically blended shade, but the pixels still use the Plus/4's existing channels.

Evening sky: Freehand Selection over the sky → **Directional Halftone / Linear** → two nearby blue shades → adjust direction and Fade → preview → Apply. You can then refine the patches using **Brush Dither** (Chapter 6). The Circular variant may be useful for a separate light spot.

Rule of thumb: If you want to fill an **area of the same visible color connected to the clicked point**, **Bucket Fill** may be the quicker option (Chapter 8). If the target is a manually drawn or intelligently determined Selection, or a complex two-channel pattern, use the tools in this chapter. The mere existence of a selection never guarantees that visible colors outside it will remain unchanged.

10. Kaleidoscope — Live Symmetrical Drawing: Free Drawing and Green Sector Only

10.1 What Is It For, and How Do the Modes Work Together?

Live Kaleidoscope creates symmetrical copies of supported Brush and Draw operations performed after it has been activated. It does not rotate the entire image that has already been drawn. Two **independent settings** must be distinguished:

Selector	Options	What does it determine?
Style	**Kaleidoscope** / **Radial**	How copies are transformed: rotation with alternating mirroring / pure rotation.
Area	**Free Drawing** / **Green Sector Only**	Which part of the original drawing operation may serve as the source for symmetrical copies.

Free Drawing therefore does not disable Live Kaleidoscope: symmetry remains active. **Green Sector Only** is not a third Style, either. Both Area options can be used with either the Kaleidoscope or Radial style. For example, *Radial* + *Free Drawing* and *Radial* + *Green Sector Only* both produce rotated copies; only the permitted source-drawing area differs.

Kaleidoscope Style: successive sectors receive geometry that alternates between mirrored and non-mirrored forms in addition to rotation. **Radial Style:** the original geometry is repeated around the center through rotation, without mirroring. The difference is easiest to demonstrate with an asymmetric, hook-shaped line.

10.2 Activation and the Actual Control Panel

Activate a Live Kaleidoscope session using the **Kaleidoscope** command at the top of the main window. The controls appear in the following visible order on the lower panel.

Label / control	Setting	Meaning
Style:	**Kaleidoscope**, **Radial**	Rotation/mirroring method; initially Kaleidoscope.
Area:	**Free Drawing**, **Green Sector Only**	Rule for the drawing's source area; Free Drawing is the default.

Label / control	Setting	Meaning
Center X:	**0–160**, step **0.5**, initial value **80**	Horizontal geometric coordinate of the symmetry center.
Y:	**0–200**, step **0.5**, initial value **100**	Vertical coordinate of the center.
Center: Canvas	Button	Sets the center to **80; 100**.
Center: Selection	Button	Sets the center to the midpoint of the active selection's **bounding rectangle**.
Center: Pick	Button	Uses the next canvas click to select the center.
Axes:	**2–32**, initial value **8**	Number of divisions in the symmetry structure.
Angle:	**−180°…+180°**, initial value **0°**	Orientation of the symmetry structure and the initial green sector.
Show axes	Checkbox, enabled by default	Displays guide axes; does not paint pixels.
Finish	Button	Ends the live symmetry session while retaining the drawing already created.

The program treats the center as a geometric coordinate consistent with the **2:1 display pixel aspect ratio**: the logical canvas is 160×200 and is displayed at 320×200. The **half-pixel X/Y increments** allow the center to lie between two logical pixels. **Selection** uses the geometric center of the smallest bounding rectangle, not the average centroid of the selected pixels. Without a selection, it does not change the center. With **Pick**, the next click chooses the center instead of starting a normal brush stroke.

10.3 Area: Free Drawing — The Original, Unrestricted Symmetrical Drawing Mode

Purpose: draw freely anywhere on the canvas while the supported tool produces multiple copies according to the selected Style. *Free Drawing* uses the original Live Kaleidoscope source-area behavior: **you do not have to keep Brush or Draw operations inside the initial green sector**. A stroke may begin in, or pass through, other sectors.

How to use it: (1) activate Live Kaleidoscope; (2) select **Area: Free Drawing**; (3) set **Style** to Kaleidoscope or Radial; (4) choose **Center X/Y**, **Axes**, and **Angle**; (5) draw with a supported Solid Brush or Draw tool at any suitable position on the canvas. The program produces repetitions based on the symmetry geometry.

Important practical distinction: if the source stroke passes through multiple sectors, individual copies may overlap one another or the original source segment. This is **not necessarily an error**: Free Drawing deliberately permits unrestricted placement. For a regular ornament built from a single initial slice, **Green Sector Only** may be more suitable. Changing Area affects the source-area handling of **subsequent** operations only; it does **not** retroactively reprocess earlier drawings.

Exercise — Free Drawing: set **Style: Radial**, **Area: Free Drawing**, **Axes: 6**, then draw a short diagonal first on the right and then on the left side of the canvas. Both can produce rotated copies; starting on the left is not rejected merely because it lies outside the initial green sector. Next, use Undo or restore a saved starting state and repeat with **Style: Kaleidoscope**: alternating mirroring changes the repetition of the diagonal.

10.4 Area: Green Sector Only — Drawing Within a Single Initial Sector

Purpose: restrict drawing to the **first, green-marked source sector**; the Kaleidoscope engine generates copies of supported operations. Select **Area: Green Sector Only** when you want to design a regular rosette, radial pattern, or mirrored ornament within just one slice.

The green sector's position and width are determined by **Center X/Y**, **Angle**, and **Axes**. The editor takes the **2:1 aspect ratio** of Plus/4 logical pixels into account, so sector boundaries correspond to the geometry actually displayed. Only portions of the source drawing that fall within the green sector are used; the remaining sectors receive rotated or alternately mirrored copies according to the selected **Style**.

What happens outside the green sector? Drawing performed there does not become a new source pattern for supported tools. The symmetrical details visible in those areas are generated from the green sector, so you do not need to redraw them in every sector.

More continuous outlines and gap prevention: when drawing symmetrical copies, the implemented engine aims to prevent missing pixels along diagonal edges, curves, and sector boundaries. It redraws the geometric shapes of the entire **Draw** tool group at their transformed target positions, rather than merely rotating the pixels of an already rasterized shape. Both **Hard Pixel** and **Anti-Aliased** edges are supported. Symmetrical painting with the Solid Brush also works with Hard Pixel and Anti-Aliased edges. Improved geometric coverage does not mean that every desired color combination can be implemented without restriction in DFLI.

Hands-on test — Green Sector: choose **Style: Kaleidoscope**, **Area: Green Sector Only**, **Axes: 8**, **Center: Canvas**. Inside the visible green sector, draw a short, hook-shaped Solid Brush stroke, followed by a thin diagonal Draw Line and a curve. Check the continuity at sector boundaries, at the center, and across more distant copies. Then change to **Area: Free Drawing** and, from the same starting state, try drawing outside the green sector: the source operation will no longer be rejected on account of the sector boundary.

Limitations: Green Sector applies only to tools explicitly handled by the engine in a symmetry-aware manner. Preventing geometric gaps and **allocating colors without hardware color clashes** are separate problems. The Area selection is session state; it does not extend the DFLI image format used in the project file.

10.5 Style and Area — Four Modes Compared

Style	Area	Drawing behavior	When is it useful?
Radial	**Free Drawing**	Purely rotated copies from any source area.	Freely positioned angular decoration; strokes started in several places.
Kaleidoscope	**Free Drawing**	Rotation with alternating mirroring from any source area.	Freer, more complex ornaments.
Radial	**Green Sector Only**	Purely rotated copies from the green source sector.	A symmetrical radial pattern built from one slice.
Kaleidoscope	**Green Sector Only**	Rotated and mirrored copies from the green source sector.	Precise rosettes and mirrored borders.

The table describes the **geometric** behavior of supported Kaleidoscope-aware drawing tools. It does not promise identical color allocation or guaranteed complete target-pixel coverage for every DFLI image.

10.6 Center, Axes, and Adjusting Live Drawing

Center: Canvas restores the geometric center to **80; 100**. **Center: Selection** aligns the center with the selection's bounding box; this applies to Rectangle and Freehand Selection alike. **Center: Pick** lets you choose the center by clicking on the canvas. Changing the center affects **subsequent** strokes, rather than geometrically rotating previously painted content.

Changing **Axes** also affects sector width: more axes produce a narrower initial Green Sector slice. **Angle** changes the initial orientation. **Show axes** is an editing guide used for orientation, not part of the finished Plus/4 graphics. **Finish** ends the live mode; it is not Undo and does not erase strokes that have already been accepted.

10.7 Supported Tools, Undo, and Color Safety

Drawing tools supported in both Live Kaleidoscope Area modes (Free Drawing and Green Sector Only):

- **Brush** → **Solid:** full support for both **Hard Pixel** and **Anti-Aliased (AA)** edges.

- **Draw → complete tool group: Line, Polyline, Smooth Polyline, B-Spline, 3-Point Curve, Box, Circle, Triangle, and Star**, including their **Hard Pixel / Anti-Aliased** drawing paths. **Redraw Shape / Draw Copy** also uses the symmetry-aware drawing behavior of Draw operations.

This support applies to both **Style: Kaleidoscope** and **Style: Radial**. Other Brush modes and non-drawing operations (such as Fill or Clipboard) **must not automatically be assumed to be Kaleidoscope-compatible**. An unsupported operation may run through its normal path, and the editor may display **"Not extended to Live Kaleidoscope mode, runs in normal mode."** Further tool-specific behavior is described in Chapters 6, 7, 9, and 12.

Hard Pixel / AA: Hard Pixel produces a crisp outline aligned to the logical pixel grid. An Anti-Aliased edge attempts to achieve a smoother appearance using available Plus/4 shades, but it is not true RGB alpha transparency. Geometrically generating each symmetrical copy does not guarantee that every intermediate color can be allocated by the hardware.

Symmetrical geometry **does not guarantee symmetrical visible colors**. **Background FF15** and **PlusColor FF16** are shared per raster line, while **Color1/AUX1** and **Color2/AUX0** are shared per 4×2 block. Smart Preserve and Origi Hardware may behave differently when collisions occur. Especially after creating AA edges or multiple shape copies, inspect the affected lines and 4×2 blocks. Save an `.mdfli` project before important drawing work; use **Undo** if necessary. It is also worth testing the Undo behavior of different operations separately.

10.8 Short Practical Workflow — Two Rosettes from the Same Stroke

1. Open a saved starting project; activate Kaleidoscope and choose **Style: Radial**, **Area: Free Drawing**, **Axes: 6**, **Center: Canvas**.
2. Draw a short asymmetric stroke anywhere on the canvas. Observe the rotated copies generated from the unrestricted source area.
3. Restore the starting image. Keep Radial Style but change to **Area: Green Sector Only**; start the stroke exclusively inside the initial green sector. Compare the restrictions on the source area.
4. Restore the starting state again; switch to **Style: Kaleidoscope**, **Area: Green Sector Only**. Alternating mirroring produces a new pattern from the hook-shaped stroke.
5. Try **Center: Selection** or **Center: Pick**, as well as **Angle**. The drawing already created should not change retroactively.
6. After **Finish**, inspect the outlines, sector boundaries, and preserved background colors at normal and enlarged zoom levels; save the project if needed.

11. Blueprint — A Reference Image Above the DFLI Canvas

11.1 What Is It For, and What Does It Not Do?

Blueprint Overlay displays an external image above the DFLI canvas as a translucent **editing reference layer**. It is useful for manually tracing the outlines of a photograph, sketch, or emblem. It **does not import the image as DFLI pixels or change the 2-bit pixel codes or the Background FF15 / Color1 / Color2 / PlusColor FF16 color values**, and the reference **is not included in PNG/PRG exports or `.mdfli` project saves**. For automatic image conversion, see **Chapter 19, Convert Image to DFLI**.

The reference image is displayed **above** the DFLI bitmap already drawn, but **below** the editor's guide grids, Brush preview, and collision markers. Hiding or clearing the Blueprint does not itself modify pixels in the actual DFLI image.

11.2 Controls in the Blueprint Overlay Dialog

Select **Blueprint** in the top row of the main window. The **Blueprint Overlay** dialog provides the following controls:

Control	Range / default	Effect
Load image...	PNG, JPG/JPEG, BMP, or GIF	Loads the reference image. The file chooser offers only supported image extensions.
Image	Loaded filename or `No blueprint loaded`	Shows which reference image is being used.
Show blueprint	On/Off; initially enabled	Shows or hides the reference layer.
Opacity	**0–100%**, initially **35%**	Visibility strength of the reference: invisible at 0%, fully opaque at 100%.
Scale	**25–300%**, initially **100%**	Further scaling of the reference, which is initially fitted to the canvas.
X Offset	**−640...+640**	Horizontal displacement relative to the initial centered position.
Y Offset	**−400...+400**	Vertical displacement.
Reset alignment	Button	Sets Scale = 100%, X Offset = 0, Y Offset = 0. It **does not** reset Opacity or reload the image.
Clear	Button	Removes the current reference image; does not erase DFLI pixels.
Apply / Cancel	Buttons	Accepts the settings / restores the Blueprint state from before the dialog was opened.

Understanding scaling: at Scale 100%, the reference is displayed at the base size fitted to the DFLI canvas while maintaining its aspect ratio. Additional Scale adjustments apply to that fitted base size; therefore **100% does not necessarily mean the source image's original pixel dimensions**. X/Y Offset values are stored in normal, 100%-view coordinates and scale together with the Canvas zoom. The image-center alignment therefore remains consistent when zoom changes.

11.3 Loading, Preview, and the Two Closing Buttons

After you select **Load image...**, the program attempts to read the image. On success, it automatically enables Blueprint visibility and shows the filename in the dialog. Unsupported or unreadable images produce an error message and do not become valid references.

Changes to Show, Opacity, Scale, Offset, Reset alignment, Load, and Clear may already appear on the canvas while the dialog is open. However, the dialog's **Apply** button is what accepts the changes. **Cancel**, or closing the dialog without applying, restores the Blueprint image, visibility, Opacity/Scale, and X/Y Offset values that were present when the dialog was opened. This restoration concerns the reference layer; it is not a DFLI editing Undo operation.

Clear removes the reference, not the finished drawing. If you press it accidentally, **Cancel** can restore the reference from before the dialog was opened; **Apply** accepts its removal.

11.4 Practical Example — Tracing a Portrait

1. Create a new image or save the current graphic in a separate `.mdfli` file. Open **Blueprint** from the top of the main window.
2. Click **Load image...** and select a portrait. Ensure **Show blueprint** is enabled.
3. Start with **Opacity 35%**, **Scale 100%**, **X Offset 0**, and **Y Offset 0**. Use Scale and Offset to align the eye, nose, and jaw contours with the desired logical positions.
4. Click **Apply**. Configure **Solid Brush** or an appropriate **Draw** tool (Chapters 6–7), then manually trace the outlines onto the DFLI canvas.

5. To inspect the completed detail, reopen the Blueprint dialog, disable **Show blueprint**, and click **Apply**. The original DFLI image is now visible without the reference.
6. If you want to use a different photograph or restart the program, reload the reference as needed: **the `.mdfli` project does not save it**.

Limitations: an external reference overlaid on the bitmap does not make it possible to create new Plus/4 hardware colors. When tracing an outline, collisions caused by DFLI's shared color carriers can occur just as they can without a Blueprint. The quality of the translucent overlay may depend on the image zoom, the source photograph's size, and display interpolation.

12. Edit — History, Selection, and Clipboard

12.1 What is Edit for, and where can you find its commands?

The **Edit** dialog brings together the history controls, selections, copying, cutting, pasting, and independent transformation of copied image regions. Its menu contains four groups. Cleanup operations are described in **Chapter 13**, detailed Undo/Redo behavior in **Chapter 24**, and Set options in **Chapter 5**.

Group	Available commands	When to use them
HISTORY	Undo, Redo	When an operation can be undone or redone.
SELECTION / CLIPBOARD	Select area, Deselect, Copy, Cut, Paste, Edit Clipboard	Operations involving a selection or the internally stored copied stamp.
CLEANUP / OPTIMIZATION	Clear GFX, ReFill Colors, Optimize PaintChannels, Clean Hidden AUX, Smart AUX Dither Prep, Dark Shade Rescue	Maintenance of image and color data; see Chapter 13.
PROGRAM SETTINGS	Set, Help / About	Display settings and information.

An important distinction: a Selection only defines the target area; Copy does not paint anything by itself, and Edit Clipboard transforms the *internal copy*. The main canvas may actually be modified by Cut/Fill or a subsequent Paste operation.

12.2 Select area — Rectangle, Freehand, and Smart Area

Edit → Select area offers three selection methods. Creating a selection does not, by itself, change the image's 2-bit pixel codes or colors.

Selection	Shortcut	What does it select?
Rectangle	**Ctrl+Q**	A rectangle defined with the mouse.
Freehand	**Ctrl+F**	An irregular area outlined by hand.
Smart Area	**Ctrl+S**	An area connected to the clicked point in four directions and having the **same visible Plus/4 color**.

How to use it: choose a selection type, select the desired image region on the canvas, then use the completed Selection for Copy, Cut, or a selection-based Fill operation. **Deselect / Ctrl+D** removes the selection but **does not delete any pixels**. When making a selection, the editor may temporarily set Brush Size to 1.

Here, Ctrl+S means Smart Area Selection, not Save. Project saving is available under I/O / Project I/O (Chapter 18). Smart Area is not object recognition: it does not automatically combine two separate patches merely because

they have the same color. A selection is not a universal boundary for every Brush/Draw operation, nor does it isolate the effects of shared DFLI color registers. Selection handling in Bucket and Fill is covered in Chapters 8–9.

12.3 Copy, Cut, and Paste — the complete workflow

Copy (Ctrl+C) stores the selected image region as an internal **Clipboard stamp**. The stamp is not a simple RGB image: it retains the transparency mask of the copied pixels, their **2-bit PaintChannel codes**, and the actual Plus/4 color information. **Paste (Ctrl+V)** activates stamp-placement mode; you then choose the destination on the canvas. Before pasting, you can also modify the stamp in **Edit Clipboard (Ctrl+E)** (Section 12.4).

Shift + Paste: this can align the stamp's placement with the **4×2 DFLI block phase**. It may reduce some hardware color conflicts, but it does not automatically resolve every color clash.

Cut — an important current difference: Edit → Cut performs Copy first, then starts the **Fill with Color** workflow for the selected area. In the source version examined, **Ctrl+X** behaves differently: it starts only Fill with Color and **does not ensure that Copy is performed**. For reliable cutting, therefore, use **Edit → Cut**, or perform **Ctrl+C → Fill with Color** as separate steps. Cut does not create a transparent hole in the DFLI canvas; valid hardware channel colors remain at that location. This is an outstanding difference in program behavior, not an intentionally equivalent shortcut.

Three Copy–Paste workflows — which should you choose?

Copy → Paste, **Copy → Edit Clipboard → Good DFLI-safe paste → Apply → Paste**, and **Copy → Edit Clipboard → Good DFLI-safe visual → Apply → Paste** are not three independent main-canvas paste engines. They differ in how the *Clipboard stamp is prepared* and in the associated **Smart Preserve / Origi Hardware** policy. Here, the name Good DFLI-safe **paste** refers to Edit Clipboard's *channel-preserving processing mode*; it is **not the same as** the **Safe Mode** button in the later collision dialog.

Workflow	What happens in the Clipboard?	Main-canvas Paste policy	Recommended use
1. Copy → Paste	The originally copied mask, Plus/4 colors, and PaintChannel codes are passed on without intermediate image reconstruction.	The policy associated with the copied stamp applies: **Smart Preserve** or **Origi Hardware**.	For **straightforward copying**, this is the most direct approach with Smart Preserve.
2. Copy → Edit Clipboard → Good DFLI-safe paste → Apply → Paste	After an **actual geometric/color transformation**, DFLI-compatible reconstruction first tries to retain the source PaintChannel if it can reproduce the color exactly.	**Smart Preserve**, according to the mode selected when Apply is pressed.	A good starting point after **Rotate, Resize, Bend, or Mirror**, when retaining the original channel assignment matters.
3. Copy → Edit Clipboard → Good DFLI-safe visual → Apply → Paste	During actual reconstruction, it first looks for an *allowed* carrier capable of reproducing the exact color, in **FF15 → FF16 → AUX0 → AUX1** order; it does not insist on the original PaintChannel.	**Smart Preserve**, using the same main-canvas engine as workflow 2.	Testing an **alternative PaintChannel assignment** for a transformed stamp; **not universally more color-faithful** than workflow 2.

An important, easily misunderstood case: if, after Copy, you **only change Mode and press Apply** in Edit Clipboard, without performing any image transformation, the program **does not reconstruct** the stamp's color or channel data. Workflows 2 and 3 therefore pass on identical image data in this case, and both select the Smart Preserve policy. If workflow 1 also starts in Smart Preserve mode, the prepared stamp may be identical; this **does not prove that the final Paste result will be identical on every destination canvas**.

Two separate checks: Edit Clipboard's **Quality** reports the **Keep / Reassign / Approx** proportions of the stamp's internal transformation. The later **Paste collisions / Safe Mode** check instead determines whether visible

surrounding and target-pixel colors can be preserved **at the chosen destination position**. A favorable Clipboard Quality score **does not guarantee** a collision-free final Paste.

Paste collisions — what do OK, Safe Mode, and Cancel mean?

The **Paste collisions** dialog may appear when pasting would also change the **actual visible color (ACT index) of pixels outside the stamp's target area**. This is because FF15/FF16 colors are shared across an entire raster line, while AUX0/AUX1 colors are shared within a 4×2 block. The program also marks the affected pixels outside the target area on the canvas. The dialog shows the number of unresolved problems and recommends trying **SHIFT + Paste** again: aligning to the 4×2 DFLI phase can shift the position by a few pixels and eliminate certain conflicts.

Button in the **Paste collisions** dialog	Action	What should you expect?
OK	You **accept the remaining color clashes**, and the program performs the full Paste operation.	In addition to pasting the target area, the visible colors of marked pixels **outside that area may change**. Under Smart Preserve, outside pixels successfully reassigned to other carriers retain their exact colors; however, OK deliberately accepts unresolved external clashes. **This is not a color-preserving choice.**
Safe Mode	The program searches for a **reduced paste mask** or a safely achievable partial plan.	Some stamp pixels may be omitted. Under **Smart Preserve**, only a separately verified partial plan may be applied: it must preserve every visible color outside the target area and, within the area, either write an exact Clipboard color or leave the original pixel untouched. If no acceptable plan exists, the image may remain unchanged. Under **Origi Hardware**, the earlier collision-mask-based Safe workflow is used; check its result separately.
Cancel	Rejects **the current paste attempt**.	The image is not modified, and **Paste placement mode remains active**. You can try another destination or Shift alignment.

There are two different types of conflict. The three-button **Paste collisions** dialog above warns that **visible colors outside the target area are at risk**. Under Smart Preserve, a separate **Paste target color conflict** warning may appear when **the target pixels being pasted** compete for the same raster-line or block color values and no fully exact solution can be found. That dialog offers **only Safe Mode and Cancel**: there is **no OK** button for approving an inaccurate target-pixel paste.

Practical decision: Safe Mode is recommended when preserving the visible colors of the existing surroundings is your priority. Choose **OK** only if you deliberately accept the **external color changes** identified by the warning. When in doubt, try **Cancel** → **SHIFT + Paste**, then compare the results. The Collision dialog and Edit Clipboard's **Quality** indicators measure different things: Quality evaluates the stamp's internal transformation, while Collision evaluates the consequences of **pasting at a specific canvas position**.

Smart Preserve Paste — preserving exact colors

The central difficulty when pasting is that **Background FF15** and **PlusColor FF16** are shared across entire raster lines, while **Color1/AUX1** and **Color2/AUX0** are shared across **4×2 blocks**. A new color for just one target pixel can therefore affect other pixels. The **Smart Preserve** and **Origi Hardware** policy associated with the Clipboard determines how hardware constraints are handled during Paste; Edit Clipboard's own Mode selection is linked to this policy (Section 12.6).

Under **Smart Preserve**, the program treats the **original visible Plus/4 colors of the copied pixels** as the desired result. It must not simply accept color-register values that happen to appear during an intermediate reassignment

as the target colors. It first attempts an entirely exact paste; to achieve this, it may rearrange 2-bit pixel codes and reallocate the per-line FF15/FF16 and per-block AUX0/AUX1 color slots.

If every target pixel can be pasted exactly and no unresolved external conflict remains, the verified plan is applied. A **target-pixel conflict** may trigger the two-button **Safe Mode / Cancel** warning, while an **external color clash** may trigger the three-button **OK / Safe Mode / Cancel** Paste collisions dialog. **Cancel** ends the current attempt without changing the image; **Paste placement mode stays active**, so another position can be selected. Safe Mode can now be attempted in some previously conflicting target cases even after a fully exact match has failed.

The Safe Mode rules are unambiguous:

- **Every visible pixel color outside the Paste target area must be preserved exactly.**
- Pixels within the target area must either receive **their original, exact Plus/4 color from the Clipboard** or **retain the visible color they had in the destination image**. Approximate color substitution is not allowed.
- The program compares several partial plans and may accept only a result verified against the complete image.

The partial search does not insist on an entirely successful image: it can preserve exactly pasteable **160×2 raster bands** and, in other bands, individually solvable **4×2 AUX cells**, while considering several possible **FF15/FF16 + AUX** assignments. Thus, a cell that can already be solved exactly is not automatically discarded because of a conflict in a neighboring cell. When several valid options exist, the verification prefers the plan producing more *new, exact target pixels*; a finalized operation corresponds to a single Undo step. If no new target pixel can be preserved, Safe Mode may finish without changing anything.

Limitation: this is a complex search over a limited set of candidates; it **does not guarantee a complete paste or a mathematically global optimum**. The preview shows the desired stamp, not proof that it can be implemented perfectly on DFLI hardware. Origi Hardware does not run the same color-preserving planner: examine the choices offered by its collision dialog and check the actual canvas result separately.

Practical test: save a copy of the project; use Freehand Selection to copy a detailed motif, then paste it onto an area with a different color-carrier arrangement while Smart Preserve is active. If a conflict is reported, choose Safe Mode. Check separately which pixels were pasted successfully, which destination pixels remained unchanged, and **the entire raster lines and blocks outside the paste mask**. Try Undo, then test Shift alignment again from the original state.

12.4 Edit Clipboard — transforming the stamp independently

Ctrl+E / Edit Clipboard opens an existing copy in a separate window. Here you can change its shape, colors, and transparency mask used by Paste. **Edit Clipboard Apply does not paste onto the main canvas:** it accepts changes to the internal stamp, after which a **separate Paste** is required. **Cancel** closes the dialog without accepting the changes.

Operation / control	Final behavior
Mirror Horizontal / Vertical	Flips the copied image horizontally or vertically.
Rotate	**−180° to +180°**; the exact 180° case uses a dedicated, pixel-identical rearrangement.
Resize	**−99% to +300%** relative to the starting size, i.e. an actual image size of **1–400%**.
Bend / Reset Bend	Deforms the image region using a **5×5 control-point mesh**; Reset Bend restores the mesh.
Smooth downscale (DFLI-safe only)	Optional smoothing during actual downscaling; off by default and inactive in **FAST** mode.
Smooth Bend (DFLI-safe only)	Optional smoothing of colors in safely processable interior Bend regions; off by default.

Operation / control	Final behavior
Transparent brush / region / channel	Marks stamp pixels to exclude from subsequent Paste; see Section 12.5.
Invert / Color filters	Affect the copied stamp's colors, not the main image directly.
Mode / Quality	The processing trade-off imposed by hardware constraints and diagnostics of the result; see Section 12.6.
Undo / Initial look	Internal undo for editing the stamp / temporary display of its original appearance.
DFLI Grid / Pixel Grid / Zoom	Affect only the Edit Clipboard preview; **Zoom 100–1600%**.
Apply / Cancel	Accept the internal copy / discard the changes.

Rotate — exact half-turn and aspect-ratio-correct rotation

A **180°** rotation of the stamp uses a special, exact pixel rearrangement: the mask, channel codes, and color data are moved directly to their reversed pixel positions. This avoids the resampling loss associated with other rotation angles. For other angles, the system uses a raster transformation adjusted to the Plus/4 multicolor pixel's **2:1 display aspect ratio**. Slanted contours can therefore still show stair-stepping or loss of fine detail.

Resize — enlargement and optional smoothed downscaling

Resize 0% preserves the original size; **−50%** targets half-size, and **+100%** targets double-size. Normal processing generally samples source pixels directly. Optional **Smooth downscale**, when enabled, uses **4×4 area sampling** only in **DFLI-safe** processing and during actual reduction: it estimates a Plus/4 palette color from the covered samples and determines the channel from the dominant source samples. This can reduce jagged edges but **may also produce approximate colors**, and it cannot preserve every detail. During enlargement and in FAST mode, this option does not perform that smoothing. The transformation workspace is capped at **240×250 logical pixels**, so an oversized or rotated stamp may be clipped at its edges.

Bend — mesh protection, edge continuity, and Smooth Bend

Bend deforms the stamp using **5×5 control points**. When control points are moved, it checks that the mesh triangles do not flip over or collapse to nearly zero area. If the requested movement is too large, it **limits the drag to the remaining valid partial displacement**. A more stable geometric assignment handles the common edges of transformed triangles, reducing accidental gaps and overwriting along their boundaries.

In a suitable DFLI-safe mode, **Smooth Bend** may estimate a Plus/4 palette color from **2×2 samples** in reliably covered interior regions. Smoothing is applied only when all required samples lie within the same triangle, inside the source image, and on nontransparent pixels; otherwise, direct source sampling is retained. The goal is smoother interior shading, **not automatic filling of transparent backgrounds or empty spaces at the mesh boundary**. Fold Guard and more accurate triangle handling **do not guarantee a gap-free result for every deformation**.

Practical test: copy a flag containing thin outlines and tonal transitions. Use the Bend handles to create a gentle wave; examine triangle intersections and motif edges. Starting from the same original, compare the results with Smooth Bend off and on. Then try reducing the stamp to 50% with and without Smooth downscale. Evaluate both the **Quality** indicators and the actual preview.

12.5 Transparency — three different exclusion rules

Clipboard **Transparency** here does not mean RGB alpha transparency. The internal mask determines **which stamp pixels take part in a subsequent Paste**.

Tool	What is excluded?	Typical task
Transparent brush	A single pixel on left-click, or a continuous pixel path when dragging with the **left mouse button held down**; intermediate points along diagonal strokes are also covered.	Removing unwanted background patches and thin diagonal outlines.
Transparent region	The four-connected region whose pixels have the same **stored Plus/4 palette index** as the clicked pixel.	Separating a single-color background.
Transparent channel	Active stamp pixels with the specified **2-bit channel code**, regardless of visible color.	Excluding an entire channel.

One complete, continuous Transparent brush stroke can be undone as a single internal Undo step. All three Transparency tools **modify only the mask**: they do not, by themselves, rebuild the colors or PaintChannel codes of the remaining pixels, even in DFLI-safe Mode. If a Rotate, Resize, or Bend preview is active immediately before a Transparency operation, its visible result is accepted first; masking does not reverse any color changes caused by the preceding transformation.

Region follows only four-directionally connected pixels; diagonally touching or separate patches of the same color require another operation. **Channel**, by contrast, may affect pixels of different visible colors if they use the same channel code. Areas excluded by the mask **do not overwrite the corresponding destination-image pixels** during Paste.

12.6 Mode and Quality — the real differences between the four processing modes

The four states of Edit Clipboard's **Mode** button are **not a progression from bad to good to best quality**. The mode determines, first, how a new DFLI color and PaintChannel assignment is built **after an actual transformation** and, second, which **hardware policy** is selected for a subsequent paste **when Apply is pressed**. The main window's Smart Preserve / Origi Hardware setting determines the **initial mode** when the dialog opens, but the Mode button can change it.

Actual label on the Mode button	Stamp preparation after a real transformation	Paste policy after Apply	When should you choose it?
Poor origi Hardware	**FAST**: carries forward the transformed result's original hardware-oriented PaintChannel data; **does not run DFLI-safe color reconstruction**.	**Origi Hardware**	When direct channel-oriented processing is intended and you are prepared to accept and examine possible color clashes.
Good DFLI-safe paste	**CHANNEL**: if a color can be reproduced exactly, first tries to preserve its original PaintChannel; if necessary, seeks another *allowed* carrier and may ultimately approximate.	**Smart Preserve**	Recommended **starting point** for transformed stamps when you want to preserve the original channel roles as far as possible.
Good DFLI-safe visual	**VISUAL**: instead of favoring the original channel, looks for an exact carrier among allowed channels in **FF15 → FF16 → AUX0 → AUX1** order; may approximate if necessary.	**Smart Preserve**	To test an alternative carrier assignment. **Not proven to be more color-faithful** than CHANNEL.
Good DFLI-safe visual FF15:$XX	Uses the VISUAL strategy but fixes the per-line **Background FF15** color to the value selected in **Set Background(FF15): $XX**.	**Smart Preserve**	When reconstruction should explicitly target a chosen FF15 background color.

When is reconstruction performed? After actual Rotate, Resize, Bend, Mirror, or certain Clipboard color-changing operations, DFLI-safe modes may build a new assignment of **per-line FF15/FF16** and **per-4×2-cell AUX0/AUX1** from visible Plus/4 colors. **Changing only Mode and pressing Apply**, without a geometric/color transformation, **is not reconstruction**: the original mask, colors, and channel codes are retained, and Apply sets the Paste policy. **Transparent brush / region / channel** are special cases: they modify only the mask and, by themselves, do not trigger reconstruction in any mode (Section 12.5).

Rules shared by CHANNEL and VISUAL: both work with the PaintChannel types actually present in the source stamp; reconstruction does not arbitrarily introduce a previously absent channel type. They seek an exactly reproducible color first, and **only when none is available** do they choose an approximate Plus/4 palette color. The two modes draw from **the same candidates for line and block colors**, but assign channels in different orders. Consequently, CHANNEL may preserve more original PaintChannels without necessarily reproducing more colors exactly. For a practical comparison of the three Copy–Paste workflows, see **Section 12.3**.

Quality — what do the three percentages measure? Keep means both the original visible color **and the same PaintChannel** are preserved exactly; **Reassign** means the original visible color is preserved **on another PaintChannel**; **Approx** means the visible color differs from the transformation reference image, has been approximated, or a covered pixel has been lost. **Keep + Reassign** together indicate the proportion of exactly preserved visible colors in the comparison. The Quality shown in FAST mode may also be a **separately calculated diagnostic assessment of DFLI-safe possibilities**, rather than the result of actually rewriting the FAST preview. A **Keep 100%** result after merely changing Mode shows that **no transformation occurred**; it does not mean that the stamp can be pasted losslessly at every destination.

The two meanings of "Safe" are not the same. Good DFLI-safe paste is the *stamp-preparation mode* in Edit Clipboard. The later **Paste collisions → Safe Mode** searches for a partial insertion plan that protects visible colors *for the particular destination canvas and coordinates*. Any Edit Clipboard DFLI-safe mode leads to the same **Smart Preserve** paste engine; however, different prepared channel assignments may produce different collisions. **Quality is not an advance guarantee of the final Paste result.**

Practical choice: (1) for straightforward copying, use **Copy → Paste with Smart Preserve**; (2) after geometric transformation, start with **Good DFLI-safe paste**; (3) if retaining original channels is less important, try **Good DFLI-safe visual** and compare the **Keep / Reassign / Approx** scores and previews. This is a **verified usage recommendation**, not a promise that one mode is better for every image or later paste destination.

12.7 Complete example, verification, and limitations

Creating a waving flag: (1) save a project copy; (2) use Select area → Freehand to select the flag, then **Copy**; (3) choose a DFLI-safe Mode in **Edit Clipboard** and bend it into a wave using the **5×5 Bend mesh**; (4) adjust the Transparency mask if needed and check the Quality scores; (5) press **Apply**; (6) start a separate **Paste** on the main canvas; (7) if a color clash is reported, consider **Safe Mode / Cancel**; (8) inspect the target area and the shared FF15/FF16 lines and AUX cells, then try Undo.

What should you watch for? Edit Clipboard **Apply** alone does not change the main image. Geometric smoothing may introduce approximate colors, while reallocating color slots for the later paste is a separate hardware-constrained task. Paste Safe Mode does not copy at all costs: it seeks the parts that can be placed exactly while protecting the visible colors outside the target area. Keep the original `.mdfli` project for important work.

13. Edit → Cleanup — Maintaining Image and Color Data

13.1 What are the Cleanup operations for?

The **Edit → CLEANUP / OPTIMIZATION** group is intended for organizing, re-preparing, or selectively correcting DFLI graphics and their color maps. It contains **six separate operations**. These are not the same as Color Filter, DFLI

Color Map, or Magic Raster under Tools (Chapters 14–17).

Operation	Main purpose	Can it modify 2-bit pixel codes?
Clear GFX	Create a completely new, blank image state.	**Yes.**
ReFill Colors	Set uniform colors for selected channels.	**No.**
Optimize PaintChannels	Find a more favorable internal color-carrier assignment.	**Yes.**
Clean Hidden AUX	Clean unused Color1/Color2 slots.	**No.**
Smart AUX Dither Prep	Prepare AUX color pairs for two-channel dithering.	**No**, but it may change AUX colors that are in use.
Dark Shade Rescue	Bring out dark details using existing color carriers.	**Yes.**

Basic rule: a pixel's 2-bit channel code is not the same as its actual Plus/4 color. Background FF15 and PlusColor FF16 are shared across raster lines, while Color1/AUX1 and Color2/AUX0 are shared within 4×2 blocks. As a result, certain Cleanup operations can visibly affect **several pixels across an entire line or block**. It is advisable to save a separate `.mdfli` project copy first.

13.2 Clear GFX — clearing the entire image

Purpose: create a clean, black starting DFLI graphic. Select **Edit** → **Cleanup** → **Clear GFX**; a warning asks whether you really want to clear the bitmap and all color maps. **Yes** performs the operation; **No** cancels it.

Data	State after the operation
160×200 pixel codes	All become **0**, i.e. Background FF15.
Background FF15	**$00**, black, on every line.
PlusColor FF16	**$71**, white, on every line.
Color1 and Color2	**$00**, black, in every 4×2 block.

Visible result: a uniformly black canvas; the white FF16 is invisible because no pixel has code 3. The actual modification can be reverted with Undo. If all data were already in the required state, a new Undo entry may not be created. **Clear GFX does not reset all of the program's interface and brush settings.**

13.3 ReFill Colors — setting entire channel color maps

Purpose: fill one or more hardware channels with the same Plus/4 color value **without changing the bitmap pixel codes**. The dialog has four rows: **Background FF15, Color1, Color2, PlusColor FF16**. Each row has a checkbox, Plus/4 color swatch, and hexadecimal color value; all four channels are selected by default. Leave checked only the channels you really want to modify, select the colors, then press **Apply**. **Cancel** discards the operation.

How it works: the two raster-line registers are written across all raster lines, and the two AUX colors are written across all corresponding 4×2 blocks. The operation **also changes currently unused, hidden AUX slots** when their channel is selected. Existing bitmap codes remain unchanged, but image regions using those channels may undergo substantial visible recoloring.

Example: suppose you want to make every Color1 detail in an image dark blue. Leave only **Color1** enabled, select a dark blue, and press Apply. Examine all pixels using the Color1 carrier across the entire image. **Limitation:** this is not Smart Preserve retouching; original shades associated with the modified slots may be lost. The image modification can be undone.

13.4 Optimize PaintChannels — the same color on a more favorable carrier

Purpose: find an arrangement of pixel channels and color registers that makes better use of the available DFLI carriers. The same visible Plus/4 color may be available on more than one channel; in that case, a pixel may **switch to another 2-bit code without any visible color change**.

The **Optimize PaintChannels** dialog offers two main options:

Option	Meaning
Automatic optimization	Automatic search for a raster-line register and pixel-channel assignment.
Prefer fixed 0 Background FF15	Favor the background color chosen with the Plus/4 color picker, with a separate **LOSSLESS ONLY** constraint.

The **0** in the second option's label identifies the Background **channel code**; it does not mean that only black **$00** can be a fixed background color. After selecting the option, **Run Optimizer** starts processing; **Cancel** exits without processing.

Interpreting the diagnostics: FF15 / FF16 changed rows counts lines whose row colors changed; **Pixel channel changes** counts pixels reassigned to another carrier; **Background in/out** and **PaintChannel usage** describe channel-code distribution; and **Visible Color Changes** counts actual visible color differences. **LOSSLESS YES / NO** classifies the color-preservation result of the optimization, not a promise about subsequent painting or Paste operations. **Visible Color Changes = 0** means that visible colors did not change in the evaluated result, even if many pixels were assigned to different channels. This **does not guarantee that future painting and Paste operations will be lossless**.

Test: save two identical project copies. Run Automatic optimization on one and the fixed-FF15 mode on the other. Compare carrier statistics and visible-color statistics, then use the **Heatmap** to examine channel distribution. On an already well-optimized image, little or no change is a natural outcome.

13.5 Clean Hidden AUX — cleaning unused color slots

A 4×2 block retains Color1/AUX1 and Color2/AUX0 color values even if **none of its eight pixels uses the corresponding channel**. Such a hidden slot may later appear as an unexpected painting color. **Clean Hidden AUX** aligns currently unused AUX values with the block's **dominant visible Plus/4 color**.

How to use it: open **Edit** → **Cleanup** → **Clean Hidden AUX**. If the program finds hidden AUX slots that can be modified, it shows the number of AUX0/AUX1 candidates and requests confirmation. If no such slots exist, it reports this without making changes. Automatic cleanup following an import may use a separate workflow. **Visible result:** the current image should remain unchanged because the operation touches only AUX slots not referenced by any pixel in the block. It does not alter pixel codes, used AUX slots, or FF15/FF16 row registers.

Example: you want to organize hidden colors in an imported image before later retouching. After cleanup, the image looks the same, but unused Color1/Color2 slots contain more consistent values. **Limitation:** do not use this operation if you intentionally stored predefined *reserve* colors in unused slots; they may be overwritten.

13.6 Smart AUX Dither Prep — local Color1/Color2 tone pairs

Purpose: prepare Color1/Color2 color slots in 4×2 blocks for **Brush** → **Dither** → **Channel Pair** (Section 6.6). This tool **does not paint a pattern**. Instead, it tries to establish a useful pair of lighter/darker colors for subsequent two-channel dithering.

The processing scope can be **Whole image** or **Selection only**. A completed selection is required for **Selection only**; without one, the operation does not start. Selection only does not restrict AUX changes to pixels actually

inside the selection: **it may examine every entire 4×2 block touched by the selection**. Because AUX colors are shared within each block, changes may affect pixels outside the selection but inside the same block.

Block condition	What does it try to prepare?	Visible color change?
Neither AUX slot is used	A reserve Color1/Color2 pair with different brightness levels, matched to the dominant tone.	Not in the current image, because neither slot is used.
Only one AUX slot is used	An appropriate partner color in the free slot.	Normally not, because the used color is retained.
Both are used but have the same color	Adjusts one channel to a slightly darker/lighter shade; prefers the channel used by fewer pixels, or AUX1 if usage is equal.	**Yes, possibly**, because an in-use channel's color is rewritten.

The confirmation analysis may report the numbers of blocks examined, hidden AUX candidates, unused slots to be modified, and **pairs of identical AUX colors that are both in use**. Pay particular attention to the last category: modifying those pairs may affect visible pixels.

Practical example: select a brown stone wall where many Color1/Color2 pairs are identical. After **Smart AUX Dither Prep** → **Selection only**, examine which blocks received new tones. Then try the two channels in **Brush Dither** → **Channel Pair**. **Limitation:** there is no guarantee that every block will receive an aesthetically suitable pair, or that separating an already-used pair will leave the image unchanged. **Clean Hidden AUX**, by contrast, deals only with hidden color slots.

13.7 Dark Shade Rescue — recovering dark image details

Purpose: distinguish dark areas that merge together by using one of the **four local PaintChannels already available**. This is **not a Gamma/Brightness filter**: it does not write new FF15, FF16, Color1, or Color2 color values. Instead, it may change certain pixels' **2-bit channel codes**. If none of the local carriers provides a suitable tone, the pixel remains unchanged.

The search considers the pixel's current brightness, color saturation, hue family, and immediate surroundings. The operation collects the proposed changes first and writes them only afterward, preventing previously modified pixels from distorting the input to subsequent decisions.

Strength	Entire image	Selection only	Characteristic
Safe	Safe Whole image	Safe Selection	Conservative; requires stronger supporting evidence from the surroundings.
Medium	Medium Whole image	Medium Selection	May intervene in more dark details.
Strong	Strong Whole image	Strong Selection	Broader candidate search; greater risk of unwanted brightening.

Portrait correction: save a project copy, select the shaded part of a face, and run **Dark Shade Rescue** → **Safe Selection**. Check whether hair and skin tones become more distinct, then compare Medium and Strong from the same *saved starting state*. **Limitation:** this method cannot restore information that was present in the original RGB photograph but lost before or during conversion; it can use only the Plus/4 colors already available locally.

13.8 Which Cleanup operation should you choose?

What do you want to achieve?	Suggested tool	What to check
Start a completely new, black image.	**Clear GFX**	Bitmap and initial color-map values.
Give all values of a channel a uniform color.	**ReFill Colors**	Effects across entire lines/blocks and on hidden AUX slots.

What do you want to achieve?	Suggested tool	What to check
Find a better carrier arrangement while preserving the image.	**Optimize PaintChannels**	Visible Color Changes, LOSSLESS, Heatmap.
Organize unused AUX colors.	**Clean Hidden AUX**	The original visible image remains unchanged.
Prepare two distinct AUX tones for Channel Pair dithering.	**Smart AUX Dither Prep**	Possible visible changes to in-use pairs.
Separate dark image details that merge together.	**Dark Shade Rescue**	Compare Safe → Medium → Strong cautiously.

These three targeted maintenance operations are not an automatic processing chain. Clean Hidden AUX, Smart AUX Dither Prep, and Dark Shade Rescue modify different data and serve different purposes. The final check must always be of the **entire visible DFLI image**, not merely the number of modified channel codes.

14. Color Filter — Basic and Advanced Color Effects

The **Tools** → **Color Filter** dialog is used to transform the color appearance of an existing DFLI image and, in certain Advanced modes, its bitmap pattern as well. It has **two separate tabs**: **Basic Filters** (14 choices) and **Advanced Filters** (6 choices). Filters are not independent effect layers that can be toggled on and off: **Apply** modifies the image's DFLI data. **Edit Clipboard** → **Color Filter** operates only on the copied stamp; see Chapter 12.

14.1 Using the Dialog and Defining the Scope

Open the dialog using **Tools** → **Color Filter** in the main window. Filters and their controls appear on the left; **Preview (2:1, 200%)** on the right displays the result at **640×400** pixels, preserving the 2:1 physical pixel aspect ratio of the 160×200 DFLI image. Changing settings updates the preview, but only **Apply** modifies the main image. **Cancel** closes the dialog without making changes.

Line range: sets the first and last raster lines to process, within **0–200**. **Start** is inclusive and **End** is exclusive: for example, **Start 40 / End 80** targets lines **40–79**. The program **aligns both boundaries to even-numbered lines** to ensure consistent handling of 4×2 DFLI blocks. An empty or reversed range (End ≤ Start) does not modify the image. **For Basic Filters, Line range is located beneath the Preview on the right. For Advanced Color Halftone, Advanced Line range also moves there; for the other Advanced filters, it remains on the left.** The compact Start and End controls appear in two stacked rows. Basic and Advanced filters use their own independent Line range settings.

Basic Effect mode: Full applies the filter to every line in the selected range; **Comb** repeats **2 active lines followed by 2 skipped lines**, counting from the beginning of that range. Comb is **not** the same as the Basic **Scanline Mood** filter, and Advanced modes have no equivalent selector. **Filter strength** is generally **0–100%**, where **0%** retains the existing colors and **100%** applies the full target mapping. This control is hidden for **User Colors**, which always maps colors using the enabled target palette.

Hardware-level distinction: most Basic filters, as well as Duotone/Tritone, Posterize, and Shade Harmony, **remap palette color indices**. They alter **FF15/FF16**, shared across raster lines, and **AUX0/AUX1**, shared within 4×2 blocks; they **do not redraw the pixels' 2-bit codes**. **CRT Scan** and **Color Halftone** are different effects that also modify bitmap codes. **Do not assume that Color Filter operations automatically protect surrounding image colors through Smart Preserve**: these are direct color-carrier/bitmap processing operations. Because carriers are shared within the selected lines, examine the entire affected image area after applying a filter.

14.2 Basic Filters — 14 Available Filters

Name shown in the program	Operation and use
Black & White with Threshold	Separates palette colors by brightness using **Threshold 0–255**: black below the threshold, white otherwise. Initial value **128**. Suitable for high-contrast, poster-like artwork.
Grayscale	Finds the nearest shades in the **2nd column** of the Plus/4 palette, together with black/white. This is not unrestricted 24-bit RGB grayscale.
Sepia	Maps colors to shades from the **8th palette column** and black, producing a warmer, vintage appearance.
Hybrid Blue	Uses black and permitted colors from the **7th, 14th, and 15th palette columns** to create a bluish appearance.
Hybrid Green	Uses black and shades from the **6th, 11th, 13th, and 16th columns** for a greenish appearance.
C64 colorized	Maps to **16 predefined Plus/4 colors** inspired by the Commodore 64's appearance. This is not a C64 file conversion.
Purred	A distinctive Purple-Brown-Red style using black and colors from the **3rd, 9th, and 12th palette columns**.
Yelleen	Black, **dark green from the 13th column**, and **light yellow from the 8th column**; a strongly defined color character.
Vivid Pastel	Uses a target set selected from the TED palette according to brightness and RGB color-distance criteria; the name does not guarantee a pale, low-saturation result.
Blue Shadow Silhouette	Uses black for dark pixels and selected bluish shades for other colors, producing a silhouette effect.
Scanline Mood	Repeats **2 normal lines + 2 darkened lines**; color carriers receive darker TED shades on certain lines, while bitmap codes remain unchanged.
Tetradic Neon	Several strongly contrasting, predefined color families for a neon-like appearance.
Tetradic Earthy	Preselected color families with an earthy character.
User Colors	**30 editable color slots**, with a starting selection generated automatically from the image; maps to the enabled colors using the nearest RGB match. Details: **14.3**.

Threshold experiment: on a portrait, start with Black & White at **128**. Near **160**, more of the original colors may turn black; near **100**, more areas may turn white. Reducing **Filter strength** is different: it blends the overall color mapping back toward the original colors rather than moving the B&W threshold.

14.3 User Colors — Intelligent Starting Palette and Manual Color Selection

The **User Colors** panel contains **30 color slots**, arranged in a **5×6 grid**. Each slot has a **checkbox** and a **Plus/4 color swatch**. The program constructs the starting palette **automatically from the current image when the dialog opens**. It determines the actually visible colors by interpreting each bitmap pixel's 2-bit code together with its corresponding **FF15, FF16, AUX0, and AUX1** values, then selects up to 30 representative TED shades.

The selection is **not random**, nor does it simply copy the 30 most frequent colors. It considers both **how often colors occur in the image** and their **RGB color distances**. If **black** is visible, the true TED black **$00** is included. If the original image contains **no more than 30 distinct colors**, all of them may be included in the starting set; remaining slots appear with their **checkboxes disabled**. Unused slots can be enabled manually later.

Manual adjustments: click a color swatch to choose another TED color from the **Plus/4 palette selector**. The adjacent checkbox determines whether that color participates in mapping. The displayed image maps the original color carriers to the **closest RGB match among the enabled colors**. Repeating a color does not increase the effective palette size. **If all checkboxes are cleared**, the original color indices are retained. Manual edits are **not lost** when Preview refreshes; automatic color selection does not restart after each click.

Important limitation: "30 colors" is the **maximum size of the permitted target palette**, **not** 30 independently available colors within one 4×2 cell. This mode does not re-optimize DFLI codes: it replaces existing FF15/FF16/AUX color values with colors from the selected set while preserving bitmap codes. Consequently, merging similar colors can eliminate details even when the palette represents the original image well.

Practical test: open a colorful portrait, select User Colors, and inspect the starting colors. Disable one of the similar shades and observe which details change; then replace that shade with a different TED color. You can also try a three-color poster: enable only three distinct colors and disable the rest. Use **Apply** only once the preview retains the necessary outlines and key colors.

14.4 Advanced Filters — Six Distinct Image-Processing Methods

The radio buttons on the **Advanced Filters** tab occupy two rows: **Duotone · Posterize · Shade Harmony** on the first row, and **Tritone · CRT Scan · Color Halftone** on the second. Only **one** Advanced filter can be active at a time. This tab has a **separate Advanced Line range**, without Comb.

Filter	Settings	Effect and main limitation
Duotone	**Shadow**, **Highlight** Plus/4 color swatches; **Balance 0–100**, **Strength 0–100%**	Brightness-based tone mapping between two selected colors, with the final result fitted to actual TED colors.
Tritone	**Shadow**, **Midtone**, **Highlight** swatches; **Balance**, **Strength**	An interpolation calculated from three tonal reference points. The Midtone swatch is visible only in Tritone mode.
Posterize	**Levels 2–8** (default **4**), **Preserve TED hue** checkbox (enabled by default)	Reduces the number of brightness levels. With Preserve enabled, the TED color family is retained; otherwise the nearest TED color to a neutral gray is selected.
Shade Harmony	**Hue shift −7…+7** (default **+2**), **Strength 0–100%**, **Preserve TED luma**, **Black protection ($00)**	Shifts the TED color family; both checkboxes are enabled by default. Does not create a new layer of freely interpolated RGB pixels.
CRT Scan	**Scanline color** Plus/4 swatch (default **$00**)	Displays **every other raster line (odd-numbered indices)** in one selected color: sets the bitmap codes to **0** and writes that color to the line's **FF15** value. Intervening even-numbered lines are unchanged.
Color Halftone	**Grayscale / Color**, **Pattern**, **Pattern grid**, **Cell size 1–8**, **Angle 0°/45°/90°**, **Black density 0–100%**, **Contrast 0–100%**	Produces a sharp, ordered black-and-color halftone by modifying both the 2-bit bitmap and the 4×2 AUX colors. Details: **14.5**.

Using Duotone/Tritone: select Shadow and Highlight colors (and Midtone for Tritone), adjust Balance, and reduce Strength if necessary. The transition does not create arbitrary new RGB colors; the program fits the result to the nearest available Plus/4 color.

Important difference between CRT Scan and Basic Scanline Mood: Basic **Scanline Mood** darkens existing color carriers in a **2+2-line pattern** while preserving bitmap codes. Advanced **CRT Scan** sets **all pixels on every other line to a single selected FF15 color**, producing stronger, more regular scanline stripes. FF16 and AUX color values are not rewritten, but 2-bit bitmap codes on the affected lines are. Choosing `$00` produces black stripes; other selections can produce blue or red scanlines, for example.

14.5 Color Halftone Pro — Sharp Black-and-Color Halftoning

Color Halftone is unusual among Advanced filters: it **does more than remap palette colors**. For each selected 4×2 DFLI block, it sets two colors: **AUX0 = true TED black** `$00`, and **AUX1 = a selected color**. Pixels within that block then use only **code 2 (black)** or **code 1 (color)**. The processed area's **FF15/FF16** values remain unchanged. In Color mode, the program selects the block's original **most frequent non-black TED color**; if frequencies are tied, it favors the brighter one. An entirely black block remains black. In **Grayscale** mode, the colored side is replaced by a light, near-white TED color.

Control	Available values / default	Effect
Grayscale / Color	Grayscale or **Color** (default)	Uses a near-white tone or an original TED color selected from the block for the light side.
Pattern	**Square 2x2**, **Bayer 4x4** (default), **Clustered Square 4x4**, **Diagonal Micro 4x4**	Ordered threshold patterns for black/color pixels: checkerboard, fine Bayer, clustered dots, or diagonal microstructure.
Pattern grid	**Pixel-aligned** (default), **Physical aspect (legacy)**	Pattern coordinate system. Pixel-aligned follows the 160×200 logical grid directly; Physical aspect samples according to the 2:1 physical pixel aspect ratio.
Cell size	**1–8**, default **2**	Scales up pattern positions: smaller values produce a finer halftone, larger values a coarser one.
Angle	**0°, 45°, 90°**, default **45°**	Pattern direction. In Pixel-aligned mode, 45° produces a diagonal arrangement on the logical pixel grid, not a geometrically exact image rotation.
Black density	**0–100%**, default **100%**	Controls coverage by black pixels together with the original image brightness. **At 0%, the filter makes no changes.**
Contrast	**0–100%**, default **50%**	Adjusts the tonal curve used to calculate black coverage from brightness. **50%** is the earlier baseline distribution; higher values may create a harder separation.

In **Pixel-aligned** mode, Angle handles the three grid-aligned orientations listed above. In **Physical aspect (legacy)** mode, sampling uses physical X/Y distances and the selected angle. Neither mode changes the image's geometric dimensions. With the same Strength/Contrast settings, the two methods can produce **different visual patterns**.

Suggested experiment: choose a multicolored image with moderate contrast, then set **Color** → **Bayer 4x4** → **Pixel-aligned** → **Cell size 1** → **Angle 0°** → **Black density 100%** → **Contrast 75%**. Compare **Contrast 50%** and **100%**, then try **Diagonal Micro 4x4** and **Clustered Square 4x4** as well. These are starting points for testing, not universally optimal settings.

Limitation: each 4×2 block has only one shared AUX1 color available for the colored side. Different original colors within a block **may merge**, and black halftoning may obscure details. The Halftone filter does not automatically preserve the processed pixels' original 2-bit channel codes and does not perform separate Smart Preserve optimization. Live Preview and Apply use a halftone plan derived from the same image, but check the overall color and detail after applying it.

14.6 Applying Filters, Checking Results, and Practical Examples

Basic workflow: (1) save a copy in `.mdfli` format; (2) open **Tools** → **Color Filter**; (3) select the **Basic** or **Advanced** tab and exactly one filter; (4) restrict the raster lines with the relevant **Line range** if needed; (5) inspect the changing **Preview** on the right; (6) choose **Apply**; (7) check outlines and raster/block colors on the canvas; (8) use **Undo** if the result is undesirable.

Three comparison exercises: (A) on a portrait, try Black & White / Threshold 128, then Grayscale and Sepia in separately saved versions; **(B)** on the same image, compare Basic Scanline Mood with Advanced CRT Scan (first `$00`, then a selected TED color) to observe the difference between 2+2 darkening and 1+1 full-line replacement; **(C)** compare Color Halftone Bayer and Diagonal Micro side by side using identical Black density and Contrast settings. Always start comparison tests from **the same saved initial project state**.

Important distinction: Color Filter / Color Halftone processes an existing full image or selected raster-line range. **Fill with Color or Pattern** → **Directional Halftone**, by contrast, creates directional patterns within a previously defined **Selection**, using two chosen channels; see **Chapter 9**. The two features do not share the same processing engine or controls.

Final verification rule: Preview is not a separate Photoshop-style effect layer; Apply changes the DFLI image. Especially after CRT Scan and Color Halftone, make sure the changes to shared raster/block colors and 2-bit codes

match your intentions. Filters do not guarantee that every previously visible color or fine detail will remain unchanged.

15. Color Presets — Five Quickly Recallable Painting Color Sets

Color Presets stores five named combinations of painting colors, which can be recalled with **F2–F6**. Each combination contains four Plus/4 palette values: **Background FF15, Color1 (AUX1), Color2 (AUX0), PlusColor FF16**. **Recalling a preset changes only the colors selected for subsequent painting operations**; it does not rewrite the existing image's FF15/FF16 raster registers, AUX color maps, or 2-bit bitmap codes.

15.1 The Five Rows and the Editing Dialog

The **Tools → Color Presets** dialog has five rows, each containing an **F2, F3, F4, F5, or F6** shortcut label, a **Name of Preset** field, four color cells, and a **Capture Current** button. Clicking a color cell opens the Plus/4 palette selector. The bottom of the dialog contains **Save Presets**, **Load Presets**, and **Close** buttons.

Button / operation	Actual result
Capture Current	Copies the four currently selected **painting colors** into the chosen preset row. It **does not sample the colors of a particular canvas pixel**.
Color cell	Allows manual adjustment of the Plus/4 palette value stored in that row.
Close	Copies the names and colors visible in all five rows into the internal preset set, then closes the dialog.
F2–F6	Loads the four painting colors from the corresponding row into the current painting-color selection. It does not recolor existing artwork.
Save Presets	Copies the five visible rows into the internal set and writes them to a separate `.medpreset` file.
Load Presets	Loads five rows from a file into the internal set and dialog fields. Does not alter the visible image.

Example: select four painting colors suitable for a night scene, open Color Presets, press **Capture Current** in the first row, and name it **Night**. In the second row, choose four warm colors and name the preset **Sunset**. After **Close**, **F2** activates Night's painting colors and **F3** activates Sunset's. This does not reset the selected PaintChannel or the Smart Preserve / Origi Hardware policy, nor does it rewrite the image's color maps: a preset is a **set of painting colors**, not an image effect or color transformation.

15.2 Saving, Loading, and the Project File

Save Presets writes the entire set of five presets to a separate `.medpreset` file. The file contains the names and the four Plus/4 color values for each preset; it contains no pixels, color maps, or DFLI image data. **Load Presets** restores a set from such a file. Color Presets data can also be stored in the supported `.mdfli` **project state**, so a separate `.medpreset` file is primarily useful for independent archiving and transfer of the color set.

Important operational detail: Capture Current immediately stores the affected row; **Save Presets** copies the entire edited set into the internal state before writing the file; **Load Presets** likewise directly overwrites the loaded rows. **Close is not a separate save-confirmation prompt**, and closing the dialog does not replace saving a `.medpreset` file.

15.3 Limitations and Data Safety

Separate error messages exist for **file-opening and file-writing exceptions**. However, the current loading validation does not reject every type of malformed content: **missing or invalid hexadecimal color fields may be**

replaced with defaults, while the load operation may still appear successful. Keep separate backup copies of important color sets and inspect all five rows of color cells after loading.

Do not confuse these features: Color Presets selects colors for future painting; **DFLI Color Map** (Chapter 17) edits color registers and maps in the existing image; **Color Filter** (Chapter 14) transforms the image's overall color appearance.

16. Magic Raster

The **Magic Raster Tool** edits the **color values on each raster line** for either the image's **Background FF15** or **PlusColor FF16** channel. It does not redraw the bitmap pixels' 2-bit channel codes. Its purpose is to create the classic demoscene “rasterbar” effect: bands of light and color that change from one raster line to the next. The effect is a **static image**, not an animation driven by raster interrupts at runtime.

16.1. The Magic Raster window — order of use

Purpose: create an ordered color gradient on specified Y lines. **Operation:** Apply writes only the selected channel's line values within the specified range, **including both endpoints**; other lines are unaffected. It does not directly write the other line channel or the Color1/Color2 block colors. Because multiple pixels may reference the same line color, the visible effect is not limited to a single drawn object.

Procedure: (1) save the project; (2) choose `Background FF15` or `PlusColor FF16` as the target; (3) set `Start` and `End` within 0–199, for example 60–100; (4) select a `Preset` **or** edit a custom pattern; (5) enable `Black 1-pixel separators between stripe colors` if needed; (6) inspect the preview color strip; (7) click Apply and check the line-color effect across the image. Apply puts reversed Start/End values into the correct order. **Limitation:** the preview color strip illustrates the per-line color table; it is not a preview of the entire DFLI canvas.

Example: for a metallic glow along the horizon, select the `PlusColor FF16` channel and a range of 50–90, then try the built-in patterns. If too many horizontal details change, narrow the range or plan a different allocation of shared color carriers. Pixels referencing the target channel may change across the **entire raster line**. The operation does not automatically alter the image's pixel pattern.

16.2. All 16 built-in presets — exact names and source pattern lengths

Note: the heights shown in the labels for **02 and 03 are reversed relative to the actual lengths of their original FULL arrays in the source code** (see the discrepancy explained below the table). The following figures are **not invented descriptions of their appearance**: they are properties calculated from the original (`*_FULL`) Java color sequences. *Distinct colors* means the number of different Plus/4 palette indices occurring in the pattern; *color transitions* counts changes between successive color values. These numbers are not an aesthetic assessment of the complete DFLI image.

Preset (as shown in the window)	Rows in original pattern	Distinct color values	Color transitions	Suggested starting point
01. C64 RasterCrime 01 (80 pixels high)	80	8	68	Try a short, dense rhythm of bands.
02. C64 RasterCrime 02 (51 pixels high)	53	8	52	Compare with preset 01 over the same range.
03. C64 RasterCrime 03 (53 pixels high)	51	7	50	Compare the previous two variants.
04. C64 RasterCrime 04 (46 pixels high)	46	7	32	Choose among the four RasterCrime variants.

Preset (as shown in the window)	Rows in original pattern	Distinct color values	Color transitions	Suggested starting point
05. C64 Genesistyle (60 pixels high)	60	7	38	Examine whether the transitions are smoother or more fragmented.
06. C64 RasterBlast 01 (60 pixels high)	60	7	34	Use as the first RasterBlast comparison.
07. C64 RasterBlast 02 (81 pixels high)	81	8	80	Choose among the three RasterBlast variants.
08. C64 RasterBlast 03 (38 pixels high)	38	6	28	Try a shorter RasterBlast pattern.
09. C64 Wavy-Rastyle (107 pixels high)	107	8	42	Examine the detail in a longer color sequence.
10. C64 Strate style (64 pixels high)	64	12	63	Experiment with a 64-line pattern.
11. C64 Rasteditor 01 (38 pixels high)	38	10	28	Try a short Rasteditor sequence.
12. C64 Rasteditor 02 (149 pixels high)	149	38	86	Try a long Rasteditor sequence.
13. C64 Rasteditor 03 (99 pixels high)	99	27	79	Try a medium-length Rasteditor sequence.
14. Plus/4 CD4 style 01 (180 pixels high)	180	9	99	A 180-line Plus/4-inspired pattern.
15. Plus/4 CD4 style 02 (180 pixels high)	180	23	103	Another 180-line CD4 variant.
16. Plus/4 CD4 style 03 (150 pixels high)	150	76	122	A 150-line CD4 variant.

The **height displayed next to a preset's name** refers to the length of the original reference pattern; it does **not** automatically constrain Start/End. In several cases, the normal preset Apply/Preview path uses a **pre-resized, 200-line** color table and maps it to the selected line interval using `Gradient / smooth`. Its result may differ from loading the original color sequence unchanged into the Pattern Editor. According to the source, these presets are **manually selected sequences of Plus/4 colors**, not pixel-exact reconstructions of the original C64/Plus/4 demos.

A discrepancy in the current program that could be corrected: in the built-in preset Preview/Apply mapping, the label `02. C64 RasterCrime 02` points to the `RasterCrime03_200` table, while `03. C64 RasterCrime 03` points to `RasterCrime02_200`. The separate **Load selected preset** operation uses the original tables, where names 02 and 03 correspond to their own `*_FULL` arrays. This is a documented inconsistency; the manual must not claim that the two paths produce the same sequence. **Suggested verification:** compare both approaches visually using the same Start/End interval.

16.3. Open 8×25 Pattern Editor — creating your own color gradient

Purpose: create a custom sequence of raster-line colors instead of using one of the 16 built-in patterns. The window provides a scrollable color table with **8 columns × 25 rows = 200 color slots**. This is **not an 8×25-pixel bitmap**, but an ordered list of color values. If the custom list is not empty, it takes precedence over the selected built-in preset.

Workflow: `Open 8×25 Pattern Editor` → optionally `Load selected preset` or `Generate into grid` → manually edit colors → choose the Pattern mode → `Refresh preview` to view the result → `Apply pattern` to

accept the custom pattern → check the main window's preview and Start/End → click **Apply** in the main Magic Raster window to write the image. With `Cancel`, the changes made in the Pattern Editor do not become the active pattern. `Clear grid` empties the editor's cells; by itself, it does not undo a raster color sequence previously applied to the image.

Load selected preset places the **original FULL list, not stretched to 200 lines**, into the cells, switches the Pattern mode to `Original / direct rows`, and adjusts the main window's `End` line to the length of the loaded list where possible. If a pattern has 80 colors and Start is 40, the suggested End is 119. If the range would extend beyond line 199 because of Start, the program caps End at 199. This avoids automatically filling the rest of the screen by needlessly repeating the final color.

16.4. Pattern — seven line-mapping modes

Mode	Operation, measured from the selected Start line	When to use it
`Gradient / smooth`	Distributes the palette colors proportionally over the entire Start–End interval.	To stretch or compress a gradient into a shorter or longer band.
`Stripe direct`	Takes colors from the list in sequence; after reaching its end, the final color is held.	A one-time sequence of bands with a held final color.
`Repeat cycle`	Returns to the first color after reaching the end of the list.	Repeating bands.
`Every 2 rows`	Holds each palette color for two raster lines, then advances cyclically.	Wider, two-line bands.
`Every 4 rows`	Holds each color for four lines, cyclically.	Bold, wide stripes.
`Mirror bounce`	Reverses direction at the end of the list, moving back and forth without duplicating the endpoint.	A mirrored highlight or wave-like rhythm.
`Original / direct rows`	Assigns one color per raster line in the original order; holds the last color after the list ends.	Preserving an original preset's line-by-line rhythm.

Worked example: custom pattern: red → yellow → white, Start 20, End 31. With `Repeat cycle`, the twelve steps across lines 20–31 run through the three colors four times. With `Every 2 rows`, there are two red lines, two yellow lines, two white lines, then the sequence repeats. With `Mirror bounce`, the sequence is red–yellow–white–yellow, then starts again. With `Stripe direct`, line 20 is red, line 21 yellow, and lines 22–31 white. These examples describe **the logical sequence of colors derived from the source**, not the appearance of the entire image.

16.5. Type, Mood, Length — generated patterns

The generator on the right side of the Pattern Editor creates the color sequence placed into the cells using the following parameters. **Type** does not select an alternative effect for the main Apply operation; it determines how the list produced by `Generate into grid` is constructed.

- **Type:** `Fade in`, `Fade out`, `Fade in-out`, `Fade out-in`, `Mirror fade`, `Pulse`. Fade in uses the specified progression of brightness; Fade out reverses it; the in-out variants arrange the progression in both directions; Pulse generates a sequence suitable for a repeating pulse. The selected **Pattern** mode can subsequently change the final line rhythm.
- **Mood:** `Gray / black to white`, `Red`, `Blue / Sky`, `Ocean / Cyan`, `Grass / Green`, `Gold / Yellow`, `Orange / Rost`, `Purple / Sunset`. These are the generator's built-in Plus/4 palette color families. Their names indicate a color mood, not automatic color sampling from an external image.
- **Length:** 1–200. Sets how many colors the generator places into the cells; it is **not the same** as the Start/End interval in the main Magic Raster window.

Practical recipe — golden metallic strip: choose `Gold / Yellow`, `Mirror fade`, `Length: 16`; click `Generate into grid`; choose Pattern: `Mirror bounce` or `Gradient / smooth`; inspect the position of the brightest point of the gradient; click `Apply pattern`; in the main window select PlusColor FF16, Start 80, End 120; then click the final Apply. This is a **starting point to experiment with**, not an optimal recipe for every image. If the color changes affect too many foreground contours, try the other channel or a narrower interval.

16.6. Black 1-pixel separators — what to expect

When the checkbox is active, **every second line is set to black (`$00`)** over the selected color pattern; the Start line itself retains its pattern color. This is a real modification of raster-line color values, not a geometrical black horizontal line drawn between pixels. For example, with Start 50, line 50 receives the pattern color, line 51 black, line 52 the pattern color again, and line 53 black. The setting acts on the final generated color table regardless of whether the pattern is built-in or custom; across a large area, it can easily produce excessive striping.

16.7. Preservation, verification, and limitations

It is advisable to try Magic Raster modifications **after saving a separate `.mdfli` project copy**, then compare the result using Undo or the saved version. This operation affects shared line colors; another drawn object using the same channel may change color as well. The **DFLI Color Map** provides a separate manual color-map editor, described in Chapter 17; its full description need not be repeated here.

17. DFLI Color Map — manual editing of raster-line and 4×2 block colors

17.1 What is the Color Map for?

The **DFLI Color Map Editor** edits the finished image's actual Plus/4 color map, not the painting color selected for the next brush stroke. **Background FF15** and **PlusColor FF16** each have a separately stored color on every raster line; **Color1 (AUX1)** and **Color2 (AUX0)** are shared within each 4×2-logical-pixel block. The editor **does not change the bitmap's 2-bit pixel codes**. Changing a color value can, however, change the visible color of every pixel referencing that carrier.

17.2 The actual two-panel window layout

On the **left side**, the **"Per-Line Colors"** panel shows **200 raster lines** (000–199), each with two separate color fields: **BG FF15** and **PlusC FF16**. On the right, the **"4×2 Shared Colors"** panel shows **100 two-line block groups** (000–001, 002–003, …, 198–199) and **40 horizontal block columns** (00–39). Each 4×2 block has a separate **Col 1** and **Col 2** color field, arranged one below the other. **Col 1 = Color1/AUX1; Col 2 = Color2/AUX0**; do not reverse this order.

In both panels, clicking a color field opens the **Plus/4 palette chooser**. There are no checkboxes for enabling or disabling channels: all displayed color fields can be edited. The raster-line and block-row displays share the same vertical scroll position; the width of the right panel also makes **independent horizontal scrolling** available. Row and column headers remain visible while scrolling. Lines separating the two-line blocks help identify the line pairs.

17.3 Go to row / Column — targeted navigation

At the bottom of the window, the **Go to row:** numeric field accepts **0–199**, and **Column:** accepts **0–39**. The **Go** button scrolls to the specified raster line and 4×2 block column. **Go does not modify any color**: it only helps you navigate the extensive color map. For example, specifying raster line 85 and column 12 takes you to the corresponding **084–085** two-line AUX block row.

17.4 Editing colors, Apply, and Cancel

Correcting a raster-line color: locate the desired Y line in the left panel, click its **BG FF15** or **PlusC FF16** color field, and choose a new Plus/4 palette value. This can affect pixels on that line that use the selected channel code.

Correcting a local AUX color: on the right, locate the appropriate **two-line block group and block column 00–39**, then select a color in **Col 1** or **Col 2**. This can affect the visible color of pixels using the same channel within the entire 4×2 block. The operation does not automatically set the other AUX slot to the same color.

The window initially holds the selected colors **in its own editable fields**. **Apply** writes the left panel's 200×2 raster-line color values and the right panel's 100×40×2 AUX color values to the image data, then closes the window. **Cancel** instead discards color selections made in the window that have not yet been applied. The main editor handles an actual Color Map change as an Undo transaction: closing without changes does not require a new Undo step.

17.5 Practical example and important limitations

Retouching a sunset: select the horizon's Y line in the left panel and change **PlusC FF16** to a suitable warm tone. After **Apply**, inspect the **entire raster line**: other objects on that line using channel code 3 may also change color. If you only want to correct a small detail on a rock, find its local **Col 1 / Col 2** block and inspect **all affected pixels in that 4×2 area**.

The Color Map is a **direct color-map editor**, not Smart Preserve retouching or per-pixel RGB painting. It does not promise automatic protection against color clashes, and changing an AUX color slot that is not currently used **will not necessarily have an immediate visible effect**. For complex work, save a separate `.mdfli` project copy first. The **Magic Raster Tool** (Chapter 16) creates automated line-color patterns, whereas **Color Presets** (Chapter 15) only selects colors for future painting operations.

18. I/O — Project I/O, PRG Import, Export, and Quick Snapshot

The **I/O** tool group provides loading and saving editable projects, three supported PRG import paths, Plus/4 export, and image conversion. **Convert Image to DFLI** is a particularly complex operation; its complete user guide is in **Chapter 19**. The *Quick Snapshot* command in the top **File** menu is also documented here. The file formats mentioned in this chapter **are not interchangeable**.

18.1 Project I/O — editable `.mdfli` projects

Purpose. Save and reopen the editable state needed to continue working on an image. The native `.mdfli` format is neither a Plus/4 program nor a simple PNG image. The current native project format is **v7**; the loader also supports older compatible project versions. A v7 project may contain optional Brush and Redraw extensions at the end of the file.

How to use it. Select **I/O → Project I/O → Save Project (.mdfli)**, specify a file name, and later restore the project with **Load Project (.mdfli)**. The Project I/O selector contains only Save and Load buttons; close the selector using its X button. Before major changes or experimenting with an unfamiliar import, it is advisable to save under a new file name.

What is preserved? The 2-bit pixel channel codes; the per-raster-line Background FF15 and PlusColor FF16 colors; the per-block Color1 and Color2 colors; and several Selection, Clipboard, and user-interface states. **History and additional state:** the program maintains up to **64 Undo and 64 Redo** image snapshots in memory, and the v7 project file can also store up to **64 of each**. **Since v2.00**, project saving has preserved additional Brush settings (for example, parameters relating to Shade Recolor/Luma, Shade Blend, Smooth, Dither Channel Pair, and Stroke Edge), as well as the **Redraw Shape descriptor of the last successfully completed supported Draw shape**. However,

Redraw Scale for Box/Circle/Triangle/Star (10–800%) is a setting of the active copy-creation session only: it **is not saved separately**, and a new Redraw session starts at **100%** after reloading a project. The stored shape geometry remains intact, but an open Redraw preview session is not saved. The **Blueprint reference image is still not part of the saved project state**.

Practical example. If you want to try a different PRG while retouching a portrait, first save the original as `portrait_original.mdfli`; after importing, save the modified state separately as `portrait_import_test.mdfli`. A PNG snapshot cannot restore the internal channel assignments.

18.2 Import PRG — the three menu options

What is a PRG? A Commodore `.prg` file generally begins with a two-byte, *little-endian* load address, followed by memory or program data. The `.prg` extension **does not tell you** whether the contents are DFLI image data, an MC bitmap, or an executable program. Choosing the correct importer is therefore essential.

The import selector offers three options:

Menu option	Expected input	What it does
Istvanv DFLI (main)	A specific Istvanv DFLI memory layout	Reads per-line and per-block colors and the 2-bit bitmap.
Multi Botticelli (MC → DFLI)	A supported MC data structure identified using the `MULT` signature	Imports multicolor image data into the DFLI editing model.
Istvanv Multicolor (MC → DFLI)	A layout recognized by the same MC decoder	Imports multicolor image data into the DFLI editing model.

18.2.1 Istvanv DFLI (main) — expected image data

The **Istvanv DFLI** importer loads `.prg` images compatible with Istvanv's DFLI viewer. This includes **DFLI PRGs generated by Istvanv's `p4fliconv` converter** and **PRGs exported by MeDFLIstvan**. The import transfers the image and color data into the editor; it does not import the viewer program contained in the PRG as editable source code.

Important `p4fliconv` settings for importing: set the **horizontal resolution to no more than 200 pixels** and **disable X-shift** in the converter. Observe these restrictions if you intend to import the resulting PRG into Brawscope. They are compatibility requirements for the imported `p4fliconv` output, not a description of the dimensions of Brawscope's 160×200 logical canvas.

Data	Memory address and size	Meaning
Background FF15	`$1A00` – `$1AC7`, 200 bytes	Per-raster-line color value.
PlusColor FF16	`$1B00` – `$1BC7`, 200 bytes	Per-raster-line color value.
Bitmap	From `$20C0`, 8,000 bytes (25×40×8)	160×200 logical pixels.
Color1 / Color2 color components	LUMA: `$4018`, `$4818`, `$5018`, `$5818`; CHROMA: `$4418`, `$4C18`, `$5418`, `$5C18`	Local color values for 4×2 DFLI blocks.

One bitmap byte contains **four 2-bit pixels**, with bit pairs ordered 7–6, 5–4, 3–2, 1–0. These memory addresses apply to the **Istvanv DFLI layout**, not to every DFLI PRG. The decoder locates the data relative to the load address; importing may fail if the required locations are absent.

18.2.2 Multi Botticelli and Istvanv Multicolor — shared MC decoder

Data	Memory address and size	Meaning
Luminance Map	From `$7800`, 40×25 useful bytes in a 1,024-byte region	Cell luminance data.
Color Map	From `$7C00`, 40×25 useful bytes in a 1,024-byte region	Cell color data.
Bitmap	From `$8000`, 8,000 bytes	MC pixels.
Identifier	`$7BFA` – `$7BFD`: ASCII `MULT`	Required format signature.
PlusColor FF16	`$7BFE`	The source MC image's **global PlusColor value**; the decoder swaps the two nibbles of the stored byte.
Background FF15	`$7BFF`	The source MC image's **global Background value**; the decoder swaps the two nibbles of the stored byte.

Multi Botticelli (MC → DFLI). The widely used Plus/4 Multi Botticelli image-data format contains *raw* image and color data, interpreted as memory beginning at **`$7800`**. The Luminance Map, Color Map, and Bitmap occupy the address ranges shown in the table, with the customary `MULT` marker at `$7BFA`–`$7BFD`. It is not a standalone display program: the file carries the image data. **Multipaint**, for example, also uses this format, so properly generated Multi Botticelli files from that application can be imported into Brawscope.

Istvanv Multicolor (MC → DFLI). This option is intended for multicolor images created with **Istvanv's `p4conv` image converter**. Such `.prg` files contain **Multi Botticelli-layout image data together with a viewer program**. The importer reads the same image-memory structure; the accompanying viewer code does not become part of the DFLI editor's image data.

A characteristic of the current decoder: it first attempts to treat the first two input bytes as a PRG load address. If the calculated address range is invalid, it interprets the file as raw data beginning at `$7800`. Consequently, it cannot be claimed that *every* raw file is recognized automatically: the `MULT` marker must appear at the correct location in the interpreted memory image. If importing a known Multi Botticelli raw file fails, check its layout and `MULT` marker.

Practical distinction: for Multi Botticelli, select a raw image/color-data file; for Istvanv Multicolor, select the PRG generated by `p4conv`, which also contains the viewer.

18.3 Export PRG + ASM + PNG

Purpose. Generate three outputs from a completed DFLI image, each intended for a different use.

Output	Use	Important limitation
PRG	Running/displaying the image on a Plus/4	Not an editable `.mdfli` project.
ASM	Assembly development and preparation for integration into a demo	Compatibility with the intended assembler must be checked against the output file.
PNG	Viewing, publishing, and quick comparisons	A 320×200 raster image; it does not preserve an editable DFLI data model.

Technical structure of the export. `PrgExporter.exportPRG` aligns the result with the **`$1001`** load address, prepares an address space extending to **`$6000`** for image storage, and fills it according to the Istvanv layout with the bitmap (`$20C0`), Background FF15 / PlusColor FF16 per-line values (`$1A00` / `$1B00`), and Color1/Color2 data

(region start addresses `$4018` – `$5C18`). The `dfli.prg` **template**, which contains **Istvanv's DFLI viewer program**, provides the display code. The exporter inserts the current image data into this template. Thus, the output PRG is not merely a bitmap-data file: it is a Plus/4 program capable of displaying the DFLI image.

ASM output format. The generated `.asm` text contains byte tables intended for **CBM PRG Studio**. The directive is uppercase `BYTE`, not `.byte`; for example, `BYTE $00,$00,$71,$1F`. Bytes are comma-separated two-digit hexadecimal values prefixed by `$`, with **no more than eight bytes per line**. The section order is `FF15_datas` (200 bytes), `FF16_datas` (200 bytes), `Bitmap_datas` (8,000 bytes), followed by the `FF14_1` – `FF14_4` tables, each with separate **LUMA** and **CHROMA** data. The exporter also writes suggested memory-address ranges as comments for the tables. The ASM is a developer-oriented data listing, **not a standalone executable Plus/4 program**. Other assemblers may require adjustments to the directive syntax.

Example of the generated syntax:

The example byte values are illustrative, not taken from a specific image export.

How to use it. Select **I/O → Export PRG+ASM+PNG**. **Choose a single output PRG file and location in the file chooser** (suggested default name: `export.prg`). The program automatically creates the corresponding `.asm` and `.png` files in the same directory; **you do not choose three separate output files**. Then verify that all three files were created. Check the PRG on a Plus/4 or suitable emulator, inspect the ASM in your intended development workflow, and open the PNG in an ordinary image viewer.

Expected result. Three representations of the same work, intended for different purposes.

18.4 File → Quick Snapshot (F9)

The **File → Save quicksnap PNG** command in the top menu and the **F9** shortcut create a **PNG snapshot**. This is useful for comparing an image with an earlier editing state or quickly sharing a preview. It is **not** an `.mdfli` project save, **not** a PRG export, and does not save the Blueprint source image.

18.5 A short, safe I/O workflow

1. **Before importing another file:** save a separate `.mdfli` backup.
2. **When importing:** select the PRG option appropriate for the source format; after an error, check the project currently open in the editor.
3. **To continue editing later:** save another `.mdfli` project; a PNG is insufficient.
4. **For distribution:** export the PRG + ASM + PNG set and check each file individually.
5. **For quick comparisons:** use F9 Quick Snapshot, but do not treat it as a project backup.

Known safety limitations of Project I/O are also discussed in **Chapter 24**. All settings and examples for **Convert Image to DFLI** are in **Chapter 19** and are not repeated here.

Special thanks to Istvanv

Special thanks to Istvanv for allowing MeDFLIstvan to use his Plus/4 DFLI PRG viewer! It was a deliberate decision **not to create yet another competing DFLI file format**. We believe it is far more important for Plus/4 DFLI images to remain, wherever possible, in a consistent, practical format compatible with other applications. Istvanv's solution provides an excellent foundation: it is well thought out, versatile, and, we believe, still has plenty of untapped potential.

The purpose of the chosen PRG format is **not to pursue the absolute highest image quality at any cost**, but to provide **versatility and practical usefulness**: graphics should be easy to incorporate into demos, games, and other Plus/4 projects. Naturally, the Brawscope editing and conversion tools still aim for the best possible visual result, while preserving the advantages of an established, shared DFLI output format.

We also want to promote **Istvanv's work and the p4fliconv application**, the **MeDFLIstvan / Brawscope** editor, and above all **the idea of a unified Plus/4 DFLI format**. We are not starting another format war; we are building together on something that already works. **One shared format, two programs, countless possibilities: everything is free, and the Plus/4 community benefits from both. Promoting each other means promoting ourselves... Wait, what!? Yes, really!** :) :)

19. Convert Image to DFLI — from Source Image Preparation to DFLI Color Assignment

19.1 Why does Brawscope have its own converter?

Over the years, several excellent image converters have been developed for Commodore Plus/4 DFLI graphics. MeDFLIstvan's built-in converter **does not claim to be the best among them**. Its advantage lies elsewhere: the source image can be adjusted in various ways, four different search strategies can be compared, and the resulting DFLI graphics can immediately be refined using Brawscope's editing tools. This eliminates the need to switch applications for every small visual compromise.

The converter **was already present in MeDFLIstvan v2.00**. The Brawscope release adds more convenient source preprocessing, adjustable RGB Range, Gamma, Color Saturation, Retro Punch, eight-band Palette Preserve Lift, and other refinements alongside the existing conversion engines. Rather than promising miracles within the Plus/4's hardware limitations, the objective is to **give the artist more choices**.

The converter does more than replace RGB colors with their closest Plus/4 palette equivalents. A DFLI pixel stores only the **2-bit code** of one of four possible PaintChannels. Background FF15 and PlusColor FF16 values are shared across a raster line, while local Color1/Color2 colors are assigned to 4×2 logical-pixel blocks. The software must therefore choose not only **which color is desired, but also which hardware channel can carry it**.

19.2 The five logical stages of conversion

1. **Load the source.** Choose a standard image file using **Browse...** (PNG, JPG/JPEG, BMP, or GIF). This is not yet a DFLI import.
2. **Crop, aspect ratio, and resize.** The source is adapted to the Plus/4's 2:1 display pixel aspect ratio and the 160×200 logical-pixel image. The display-mapping process may also use an intermediate 320×200 image.
3. **Source preprocessing.** RGB Range, Gamma, Saturation, and Retro Punch shape the image's appearance **before the search for a DFLI color assignment**. The eight-band Palette Preserve Lift can help retain different Plus/4 luminance bands.
4. **Search for DFLI colors.** The selected Fast / Medium / Good / Best engine searches for per-line FF15/FF16 and local AUX colors, then assigns 2-bit pixel codes to them. This is where the quality modes differ technically.
5. **Accept the result. Convert** calculates the DFLI image, which can then be transferred to the editor's canvas. The **Source Preview is not the finished, hardware-assigned DFLI image**: inspect the final result on the canvas.

19.3 Resizing: Nearest Neighbor and Smooth Average

In the older v2.00 workflow, direct sampling—**Nearest Neighbor**-type image processing—was predominant: a destination pixel received the color of one source pixel. This is fast and can work well for sharp, already pixelated graphics, but fine gradients or thin contours in photographs can lose detail.

The current **Smooth Average** creates the 160×200 logical sample from the intermediate 320×200 image using **area-coverage-aware averaging**. It does not simply copy the left or right source pixel. It can produce more stable tones in dense color transitions, but may slightly soften extremely thin pixel-art lines. The key distinction is that **Smooth Average is not, by itself, an edge-detection or line-enhancement algorithm**: it does not contain

separate “edge preserve,” “detail preserve,” or “line-art hardening” stages. Edge-aware decisions appear later in the pixel-code scoring of certain conversion engines (19.6).

19.4 Source Preprocess controls introduced since v2.00

The **RGB Range Min/Max, Luminance Gamma, Color Saturation, and Retro Punch sliders** described here were not available in v2.00. They shape the brightness and color character of the photograph before the actual DFLI search. This matters because information already lost at this stage—for example, shades merged into one flat area—generally cannot be reconstructed later by the four-channel color assignment.

Control	Range / initial value	Meaning and effect
RGB Range Min	−50…100 / **−2** (−0.02)	**Range** means the tonal interval; Min is its lower limit. Controls the lower end of the brightness calculated from the source RGB colors. Lower values may retain more shadow headroom; increasing it can push dark tones into black sooner.
RGB Range Max	0…200 / **103** (1.03)	The upper end of the brightness range. Reducing it can make the image reach white sooner; increasing it leaves more highlight headroom. Min and Max **jointly** determine the luminance-range mapping.
Luminance Gamma	25…400 / **100** (=1.00)	**Luminance** is brightness; **gamma** is a nonlinear tone curve. It raises or lowers midtones while affecting the endpoints less directly. A value of 1.10 may produce brighter midtones; 0.90 may make them darker.
Color Saturation	0…300 / **100** (=1.00)	Adjusts the distance between the grayscale component and the colored components. Expect a grayer appearance at 0, a more natural one around 100, and more vivid colors above 100.
Retro Punch	0…100 / **0**	“Punch” means a more striking, emphatic visual character. The program moderately increases midtone brightness contrast and color saturation, potentially creating a bolder, poster-like retro look.

What does RGB Range actually do? Despite its name, the current implementation **does not** provide independent minimum and maximum settings for the Red, Green, and Blue channels. It first calculates weighted luminance from RGB (`0.299×R + 0.587×G + 0.114×B`) and normalizes it to the interval between **Min and Max**. Gamma then transforms the resulting luminance; the original color differences are added back relative to the adjusted luminance, taking Saturation and Retro Punch into account. Brightness and color character can therefore change together, but this is not a conventional per-channel RGB Levels panel.

Gamma in more detail: the software applies the mapping `luminance^(1/gamma)` to normalized luminance. Consequently, **Gamma > 1** can raise midtones, whereas **Gamma < 1** can darken them. A face in a dark portrait can thus be brought out without simply adding the same amount of white to the entire image. Excessive correction, however, can produce flat tones or an overly bright face.

Color Saturation in more detail: the software approximates chroma from the RGB distance between the pixel and its calculated luminance. The adjustment is not merely a fixed multiplier: the code also contains a **saturation curve**, which is modified by Retro Punch. Thus, 100 is a useful neutral starting value, but not necessarily a mathematically pixel-identical bypass of the complete preprocessing chain.

Retro Punch in more detail: this is neither a single contrast slider nor a new dithering mode. In the midtones, it affects luminance deviation by an additional factor of **up to approximately 10%**, and chroma by **up to 25%**; it also modifies the saturation curve. It can therefore make a face or colored lettering stand out, but a high Retro Punch setting may merge shades just like other strong preprocessing effects.

Suggested experiment: begin with the defaults (Min −2, Max 103, Gamma 100, Saturation 100, Punch 0). For a dark portrait, change only Gamma to 110 at first. For bright highlights, test a small adjustment to Range Max

separately. Then use Saturation and Retro Punch to shape the atmosphere. **Change only one slider at a time** so you can identify which adjustment helped.

19.5 Palette Preserve Lift and fixed per-line colors

Palette Preserve Lift offers eight separate **Band 0–7** controls, each adjustable from **−30 to +30**. It does not add eight new Plus/4 colors; instead, it influences how different brightness levels are retained or supported during source preparation. This can help bring out a shade that would otherwise disappear in a dark region, although excessive lift may change the overall appearance in an unwanted way. The Band sliders update the **Source Preview**; they **do not automatically run the complete DFLI conversion again**.

The **Neutral / White Guard** checkbox (off by default) attempts to reduce unwanted color shifts in bright, nearly neutral regions. It is particularly worth comparing on and off for white clothing, gray clouds, or highlights on skin.

The **Use fixed per-line Background FF15** and **Use fixed per-line Plus-Color FF16** checkboxes let you constrain the two shared per-line color sources to selected Plus/4 palette values. This can be useful for a deliberate graphic design, but it **reduces the search space**: the converter has less freedom to explore alternative, potentially more accurate per-line color pairs.

19.6 The four conversion engines — actual technical differences

The four quality levels are **not merely the same algorithm with different waiting times**. All four search for a solution within the Plus/4 palette and DFLI structure, but employ different starting-state, candidate-selection, and refinement strategies.

Engine	Color-assignment refinement	Difference and practical significance
Fast – V31	One selected initial two-line state, **up to 2 refinement passes**	The faster, simpler search path. It **does not include the final edge-aware, neighbor-code scoring path in the form used by Medium/Good/Best**. Useful for an initial look at colors and composition.
Medium – V35	**Up to 2 refinement passes**, taking the previous two-line state into account	It differs from Fast **in more than the pass count**: it also weights continuity between two-line colors, uses edge-dependent penalties/rewards to stabilize neighboring pixel codes, and may retry local AUX candidates.
Good – V36	**Up to 3 refinement passes**, with a larger candidate set	Some steps retain **up to 8** candidates from raster-pair combinations and use a **candidate limit of 10** in local AUX-color searches. Compared with Medium, this is not merely "one extra pass": it can consider more alternative color assignments.
Best – V38	**Up to 3 refinement passes per tested starting state**	Allows **up to 4 different starting states per two-line band** to undergo detailed refinement, then selects among them by score. This makes it possible to compare more alternative color solutions.

What is a "two-line state"? In DFLI, local Color1/Color2 colors are shared within 4×2 blocks, so the color assignments of two adjacent raster lines are linked. The program evaluates initial **FF15/FF16 raster-line color pairs** and block colors, then looks for the least erroneous pixel-code assignment for them. If hair and facial details compete for several color values within the same region, one starting state may preserve the face better while another may better preserve the hair outline.

What exactly does Best's "4 × 3" search mean? 4 is the **maximum number of starting states** retained for each two-line band, and **3** is the maximum number of refinement passes applied to each starting state. Thus, **up to four starting variants can each be refined in up to three passes**. This does *not* mean twelve complete image conversions, nor does it promise that every two-line band actually contains four distinct states and three passes that produce changes. Iteration can stop early if there is no improvement.

What role does edge detection play? In their final pixel-code selection, the Medium, Good, and Best engines examine source-image color differences **to the left and above**. They estimate horizontal and vertical edge strength

from RGB color distances. **In uniform regions**, they may prefer retaining a neighboring pixel's channel code to avoid unnecessary color noise. **At stronger image edges**, they place less weight on matching the neighbor's code because retaining the contour may matter more. This is genuine edge-aware scoring, **not a separate Sobel/Canny edge filter and not part of Smooth Average**. One expression in the code, `edgeBoost * 0.0`, has no effect; the active contributions come from flatness scoring and `edgeAwareCodePenalty`.

Why might Best not look better in every small detail? The scoring function evaluates the error of the complete affected two-line color and channel state. If one solution preserves a face better while another better preserves an especially important eye contour, the more global score may not favor the same result as the user. More trials provide a wider search, **not a guarantee of a mathematically optimal solution**.

Technical acceleration: the engines include pixel-color-error lookup tables and caches for previously solved block configurations. These reuse calculated values; they are not separate color-correction filters. Actual processing time depends on the computer and the image, so this manual does not promise particular completion times.

19.7 Practical workflows — portrait and pixel art

Portrait: load a photograph → check cropping and the 2:1 display aspect ratio → adjust Range/Gamma in the Source Preview → fine-tune Saturation/Retro Punch if necessary → experiment with Lift bands and Neutral / White Guard → compare Fast, Medium, Good, and Best outputs using identical settings. Examine the preservation of eyes, individual hairs, and shadowed facial areas, as well as the overall image colors. Save the accepted DFLI image as an `.mdfli` project, then refine it with Brush, Draw, or DFLI Color Map if required.

Already-pixelated, Plus/4-style graphics: before applying strong Retro Punch or Gamma changes, try to preserve the original color values. Smooth Average can average away tiny contours, so inspect the result especially carefully at 100% logical resolution. More computation does not guarantee a more attractive image here either.

19.8 Converter controls and accepting the result

Interface element	Purpose
Browse... / Source Preview	Select an external image and inspect source preparation.
Quality: Fast / Medium / Good / Best	Select one of four distinct conversion engines. The current default is **Best**.
RGB Range Min/Max, Luminance Gamma, Color Saturation, Retro Punch	Adjust source brightness and color character; see 19.4.
Palette Preserve Lift Band 0–7; Neutral / White Guard	Fine-tune brightness bands and neutral colors; see 19.5.
Fixed per-line FF15 / FF16	Constrain the search to a specified raster-line register color.
Convert	Start the actual search for a DFLI color assignment.
Cancel	Close the relevant dialog or request cancellation of a running conversion. Cancellation may take effect at the algorithm's designated checkpoints.

Do not confuse updating the Source Preview with running Convert. After changing a slider, first inspect the preprocessed RGB image; then use Convert to calculate the actual DFLI result. Once the image reaches the canvas, the shared raster-line colors, 4×2 AUX slots, and 2-bit channel codes determine its actual appearance. Project saving and Istvanv-compatible PRG export are separate operations described in **Chapter 18**.

In summary: Brawscope's converter is not designed to defeat competing converters, but to provide a **convenient, deliberate workflow between image conversion and editing**. Fine-tuning the source image and comparing the four search engines offer many possible starting points—but the quality of the final DFLI image must always be judged by the image itself, not by the name of the quality setting.

20. PaintChannel and Palette — Selecting the Painting Color and Identifying a Pixel's Actual Color Carrier

Brawscope's color handling requires a distinction between two things: the **painting color selected for the next operation** and the **PaintChannel code already stored for a pixel in the image**. Every pixel in Plus/4 DFLI uses one of four possible color carriers. Two pixels appearing to have the same color does not necessarily mean they use the same channel.

20.1 The Four PaintChannels and Their Hardware Scope

Pixel code	Name used in the interface	Color carrier	Which pixels may share it?
0	Background FF15	Background color for each raster line	All code-0 pixels on that line
1	Color1	AUX1	Code-1 pixels within the same 4×2 DFLI block
2	Color2	AUX0	Code-2 pixels within the same 4×2 DFLI block
3	PlusColor FF16	PlusColor for each raster line	All code-3 pixels on that line

A **PaintChannel** is therefore a hardware carrier for a color, not an RGB color range or a separate painting layer. The shared nature of these color sources explains *color clash*: changing a shared register or AUX value can also change the visible color of other pixels. The limitations and their handling by Smart Preserve are described in Chapter 22.

20.2 Mouse Actions on the Palette

The **Plus/4 Color Palette** on the right is used to set the palette colors for subsequent painting operations. **Clicking the palette does not, by itself, recolor the existing DFLI image.**

Action	Result
Left-click on the palette	Sets the **painting color** for the actually active PaintChannel, including any temporary override.
Right-click on the palette	Specifically sets the **PlusColor FF16 painting color**, regardless of the normal channel selection.
Alt + left-click on the canvas	Eyedropper: samples the visible Plus/4 color under the cursor and its associated channel.
Hold 0 / 1 / 2 / 3	Temporarily overrides the PaintChannel for supported painting operations.

When selected from the palette, black palette values (`$00` , `$10` , `$20` , … `$70`) are normalized to **`$00`** , for all four painting channels and for right-clicking the palette as well. This rule applies to palette selection; it does not mean the program automatically rewrites every previously loaded image value.

20.3 Right-Clicking the Canvas — Cycle PaintChannel

Right-clicking the canvas is not the same as right-clicking the palette. In supported normal Brush modes, the right mouse button cycles the **2-bit PaintChannel codes** of the targeted pixels:

0 Background FF15 → 1 Color1 → 2 Color2 → 3 PlusColor FF16 → 0 Background FF15.

Cycling does not create a new palette color. Instead, it redirects a pixel to another color carrier already available at that position. Consequently, the pixel's **visible color may change** if the new channel contains a different color. Conversely, switching between two channels containing the same color may change the code without any visible difference. Cycling may affect the entire stamp determined by **Brush Size and Brush Shape**. To inspect a single logical pixel, use **Brush Size = 1**.

Tool-dependent exceptions: this normal right-click cycling is disabled in the **Shade Blend** and **Smooth** Brush modes. With an active Draw tool, right-click may have another function, such as finishing a Polyline or canceling a Curve. After a right-click operation, check the actual PaintChannel code and visible color in the Pointer panel.

Example: select a point on the black outline of an image with Size set to 1. Use Pointer to check whether the pixel originally uses Color1. Cycle its channel with one right-click, then check Pointer again. If a different color appears, it comes from the new channel's local color value. Undo can restore the modification if necessary.

20.4 PaintChannel, Pointer, and Color Selection — Where to Check What

The **PaintChannel panel** controls the target channel for the next painting operation and the **Smart Preserve / Origi Hardware** behavior mode. In contrast, the **Pointer panel** reports the actual data of the **existing pixel under the cursor**: Actual X/Y, DFLI Column/Row, actual color, and Actual PaintChannel. Its detailed layout is given in Section 21.3.

Selecting a painting color does not, by itself, change the FF15/FF16 raster-line colors or Color1/Color2 block colors already stored in the image. Use **DFLI Color Map** (Chapter 17) to edit those values directly, or the **Cleanup** tools (Chapter 13) to optimize color assignments.

20.5 Removal of the Current Rasterline Y Control

The former manual **Current rasterline Y** control has been removed from the v2.64 interface. With independently selected painting colors, **you no longer need to choose a raster line in advance** to set the color of the next Brush stroke. The position where painting takes place is determined by the tool and the mouse action on the canvas. **DFLI's raster-line-based color handling remains fully in place**: only the previous, cumbersome step for setting a painting color has been removed. The cursor's Y coordinate is still available in the Pointer panel.

21. Zoom, Guide Grids, Pointer, Heatmap, and Status Bar

The main window's display and inspection tools support precise editing of DFLI images. **Zoom** adjusts the view size, the **grids** show pixel and cell boundaries, **Pointer** reports data about the pixel under the mouse, **Heatmap** visualizes use of the active PaintChannel, and the **Status bar** indicates the editor's current state and task. By themselves, these aids **do not rewrite DFLI pixels or color registers**. Their overlay colors are configured under **Set** (Chapter 5), while painting colors and PaintChannels are explained in **Chapter 20**.

21.1 Zoom — Magnifying the Working Canvas

The main canvas has a Zoom range of **100–975%**. Use the **Zoom %** slider near the bottom of the editor to change the view size. Over the canvas, the **mouse wheel** zooms in or out in increments of 25 percentage points. **Page Up / Page Down** also adjust zoom. The current value is shown in the Status bar's **Zoom** field.

The underlying image remains **160×200 logical pixels**, with the Plus/4 multicolor pixel's **2:1** aspect ratio. Zooming does not create new pixels or alter the image data exported to PRG/PNG. High zoom is useful for pixel-accurate work, while lower zoom is useful for assessing the composition as a whole.

Exercise: zoom in on a diagonal outline, turn on Pixel grid, and inspect any incorrect pixels. Then zoom out and check whether the correction also looks right in the full image.

21.2 Guide Grids — Pixel, DFLI, and Character

The three grids are toggled with separate keys. Grid lines are **editing guides** drawn over the canvas; they are not part of the bitmap and are not included in graphics exports.

Key	Grid	Division	When is it useful?
G	**Pixel grid**	160×200 logical pixels	Inspecting individual pixels, outlines, dithering, and anti-aliasing
D	**DFLI grid**	40×100 units, each measuring 4×2 logical pixels	Inspecting shared Color1/Color2 block colors and block boundaries
Caps Lock	**Character grid**	40×25 character cells	Viewing the classic Plus/4 character-cell layout

Grid color and transparency are adjustable through **Set → Pixel grid color... / DFLI grid color... / Char grid color...** (Chapter 5). Changing a guide color does not automatically turn the grid on.

21.3 Pointer — What Is Actually Under the Mouse?

The **Pointer** panel beneath the palette shows the logical position, visible Plus/4 color, and stored PaintChannel code of the pixel under the mouse. **These are not the same as the painting color selected for the next Brush stroke.** The panel has two rows and four columns:

Column	Top row	Bottom row
1	**Actual X** coordinate	**Actual Y** coordinate
2	**DFLI Column**	**DFLI Row**
3	**Actual color + hex code:** label	**Actual PaintChannel:** label
4	Swatch and hexadecimal value of the current image color	Actual PaintChannel name / value of the pixel

Actual X/Y are coordinates in the 160×200 logical pixel grid; **DFLI Column/Row** identifies the 4×2 DFLI block. Pixels with the same visible color may still have different PaintChannel codes, which Pointer makes easy to verify.

Exercise: move the pointer over two adjacent pixels of the same color. Compare their **Actual color** and **Actual PaintChannel** values. Matching colors alone do not prove that the pixels use the same carrier.

21.4 Heatmap — Visualizing the Active PaintChannel

Press **H** to toggle the Heatmap view on or off. When enabled, the program highlights with a translucent color **all pixels whose 2-bit code matches the currently selected PaintChannel**. You inspect one channel at a time: switching the PaintChannel from 0 to 1 or 2 changes the highlighted pixels accordingly.

The overlay color and Alpha value are adjustable in **Set → Heatmap color...**. This is purely a diagnostic display: **it does not paint or recolor anything and is not included in the exported image**.

Exercise: turn on Heatmap (**H**) and select the **Color1** PaintChannel. Examine the pixels using channel 1, then switch to **Color2** and compare the distribution of the two AUX carriers. Pointer also shows the actual color of each inspected pixel.

21.5 Status Bar — Mode, Zoom, Heatmap, and Actual Task

The main window's lower **Status bar** displays five pieces of information about the screen's current state and the editor's working mode.

Status field	Meaning
Mode	Current hardware-handling policy: **Smart Preserve** or **Origi Hardware**.
Zoom	Current main-canvas magnification, as a percentage (**100–975%**).

Status field	Meaning
Heatmap	**ON** or **Off**: whether the diagnostic overlay is enabled.
Heatmap channel	When enabled, the selected channel: **0-BG**, **1-C1**, **2-C2**, **3-PC**; when disabled, **None**.
Actual task	The currently active editing mode, or briefly the name of a one-shot action just performed.

Heatmap status indication. When Heatmap is active, the **"ON" label in the Status bar pulses visually**, alternating between two interface colors. When Heatmap is disabled, **"Off"** appears and the pulsing stops. This indicator draws attention to the active Heatmap view; it does not animate the canvas content.

Actual Task — feedback on your work. By default, the **active Brush mode** is shown; at startup, for example, `Brush Solid`. When you change modes, the text follows the actual selection, such as `Brush Pattern …`, `Brush Dither …`, `Brush Recolor …`, `Brush Shade …`, `Brush Luma …`, `Brush Shade Blend …`, or `Brush Smooth …`. Some modes also display the pattern name, target shade, Strength, or Recolor source and target color values. If Recolor is waiting for a source color to be picked, the label indicates this separately.

One-shot actions. After **Undo**, **Redo**, **Copy**, or another one-shot command, the Status bar may temporarily show the action's name. After approximately **1–2 seconds**, this message returns to the actually active Brush mode. By contrast, the name of a tool state that remains active is not a message that disappears after five seconds: it continues to describe the current working mode until the program switches to another state. This lets you distinguish **what you are currently working with** from **which short action just took place**.

Exercise: select Solid Brush, then Pattern Brush, and observe the **Actual task** field. Perform an **Undo**: after the brief action indication, the selected Brush mode appears again. Next, press **H** to enable Heatmap and check the pulsing **ON** label. These indicators provide separate information: one identifies the working mode, and the other shows the state of the diagnostic view.

Related chapters: guide colors: 5; Palette and PaintChannel: 20; Smart Preserve: 22; keyboard shortcuts: 23.

22. Smart Preserve and Origi Hardware — Operation, Color Clash, and Limitations

What exactly does "Preserve" mean here?

Smart Preserve aims to retain the **visible colors of pixels outside the target area**. To achieve this, it may change a pixel's 2-bit channel code or rearrange the shared color data that carries its color: the **same visible appearance** does **not** necessarily mean identical internal DFLI data. A *Reassign* result can therefore be visually correct. By contrast, the hardware-oriented *Origi Hardware* path prioritizes direct changes to channels and registers, which may also affect other pixels sharing those color sources.

Practical check: save the same starting image as two separate project copies. At the same location, try the same painting color in both modes, then compare (1) the color of the targeted pixels, (2) any color changes in surrounding, non-targeted pixels, and (3) the channel assignments reported by Pointer/Heatmap. **Limitation:** Smart Preserve does not guarantee a lossless, hardware-compatible solution in every situation; some operations may be rejected, produce partial results, or require separate conflict handling. This comparison concerns **editing modes**, not the image conversion described in Chapter 19.

The problem: painting one pixel can damage others

Imagine a portrait with a dark-blue background: you want to draw a thin, light-colored outline beside the face. In a conventional RGB editor, you would change only the local pixels of that outline. In DFLI, however, the outline might use a shared Background FF15/PlusColor FF16 raster-line color or an AUX block color. Changing that color can also recolor **non-targeted parts** of the portrait. This is known as **color clash**, and it may require subsequent manual correction.

What does Smart Preserve do?

Smart Preserve is one of the most important improvements in v2.64: a hardware-aware painting policy that analyzes channel assignments **before** the intended painting operation in supported Brush, Draw, Fill, and Paste workflows. It can attempt to move 2-bit pixel codes to other channels that carry the same visible color and to re-plan the **Background FF15/PlusColor FF16** values shared by each raster line and the **Color2/Color1** values shared by each block.

The goal is not merely to warn about a clash, but to actively search for a DFLI-compatible solution that preserves the existing visible image outside the selected painting area. This allows many color clashes to be resolved automatically, or their visible effects to be reduced, within the limitations of the Commodore Plus/4 DFLI format.

Example — an outline beside an eye: two pixels with logically identical visible colors can be represented using different hardware channels. If one channel can be freed for a new line color, Smart Preserve may reassign the existing pixels to another carrier. This *channel reassignment* is the central idea behind Smart Preserve; the outcome always depends on the actual local register allocation.

Clipboard Paste: Safe Mode and partial maximization

Smart Preserve for Clipboard Paste is particularly complex because **Background FF15 / PlusColor FF16** values are shared across raster lines, while **Color1 / Color2 (AUX1 / AUX0)** values are shared within 4×2 cells. In addition to searching for a complete, exact paste solution, Smart Preserve Safe Mode searches for **complete two-line bands, independent AUX cells, and partial combinations of shared FF15/FF16 and AUX assignments**. Safe Mode has a strict acceptance rule: the visible colors of non-targeted pixels must remain unchanged, while target pixels must either receive their exact copied ACT color or retain their original color. Every new partial-search stage evaluates the canonical target image against the original clipboard ACT data.

Selecting the best known plan **does not prove that a global optimum has been found**: the raster-register search operates on a finite, restricted set of candidates. With the **Safe Mode / Cancel** choice, the user decides whether to accept the verified partial paste. Detailed operation and the search stages are explained in **Section 12.3**.

Origi Hardware: direct hardware behavior

Origi Hardware favors more traditional, direct handling of colors and channels. Its advantage is more predictable, direct editing of hardware data; its drawback is that changing a shared color can also affect neighboring pixels. It may be the appropriate mode when deliberately creating color-clash effects, producing specialized hardware effects, or performing very precise register-level painting.

Limitations, collision warnings, and Safe Mode

Smart Preserve **does not promise perfect, lossless results for every operation**. Different Brush, Fill, Paste, and Draw paths provide different collision-handling options. If the four desired visible colors cannot be represented within a particular block or raster line under the hardware constraints, the program may warn you, skip problematic pixels, offer a Safe Mode procedure, or use another tool-specific path. **Anti-Aliasing** may also require additional colors or shades along edges, making it especially sensitive to the available channels.

Simpler color handling: "Less Coding, More Creating"

Version 2.64 improves more than the underlying algorithm. The interface distinguishes more clearly between the **color selected for painting** and the **hardware color registers actually stored in the image**. Labels such as *Background, Color1, Color2,* and *PlusColor* make everyday work easier; the Background FF15/PlusColor FF16/AUX values remain accessible in advanced tools. Artists therefore do not have to think in terms of TED registers with every brushstroke—but hardware-level control is still available whenever needed.

Three possible outcomes for the same drawing request

Resolvable conflict: the required background color is already available on another carrier channel. The planner may redirect existing pixels to that channel, freeing a slot for the new color. Result: the intended drawing can appear without changing the visible colors of surrounding pixels.

Partially resolvable conflict: only some sections of the requested line can be represented. In that case, you must decide through the tool's own collision-handling or Safe Mode interface. A partial result does not mean that the entire intended brushstroke was completed losslessly.

Unresolvable conflict: the four channels and existing colors in the relevant line or block cannot accommodate the requested new color without other changes. Depending on the tool, the program may display a warning, skip pixels, or use another processing path. This can follow from *the finite capacity of the format itself*; it is not necessarily an algorithmic defect.

23. Keyboard and Mouse — Quick Reference

How do keyboard shortcuts work in practice?

Many shortcuts in the main window are bound to its root panel, so they may work while different controls are active **within the main window**. Some keys used for drawing and diagnostics on the canvas, however, are handled by the **Canvas's own keyboard handler**. If `H`, `P`, `L`, `0`–`3`, `Page Up`, or `Page Down` does not respond, first click back on the drawing canvas: text-entry fields and separate dialogs may intercept the key. Pressing a shortcut does not automatically mean that the image will change; many commands only select a mode, open a dialog, or toggle a temporary guide view.

First practical test — selection and Bucket: pressing `Ctrl+Q` starts Rectangle Selection, and the program sets the **brush size to 1 pixel** for selection work; the other brush settings do not necessarily change. Select an area, then press `Ctrl+B` to open the Bucket settings. If you do not yet want to fill anything, close the dialog without applying it. `Ctrl+D` clears the selection. Here, `Ctrl+S` means **Smart Area Selection**, not project saving: use I/O → Project I/O to save a project (Chapter 18).

Second practical test — temporary channel override: select a supported drawing tool, make sure the canvas has keyboard focus, and **hold down** `0`, `1`, `2`, or `3`. This sets the internal paint index to the corresponding value: `0` Background FF15, `1` Color1, `2` Color2, `3` PlusColor FF16. The override ends when the key is released. This is not the same as permanently changing the PaintChannel panel; the particular drawing mode determines how it uses the temporary override. Holding the key down does not paint anything by itself.

Third practical test — precise horizontal and vertical strokes: on the Canvas, holding `P` activates a **vertical** direction lock, while holding `L` activates a **horizontal** direction lock on the affected painting paths. Draw a short stroke without a lock, then repeat while holding the relevant key. The locked stroke should follow a single X or Y direction; the tool's geometry and DFLI color limitations still apply. Releasing the key disables the lock.

Fourth practical test — zoom: when the pointer is over the canvas, the mouse wheel adjusts the Zoom slider in **25-percentage-point steps**: scroll up to zoom in and down to zoom out. The code limits this operation to **100–975%**. `Page Up` and `Page Down` also request 25-percentage-point adjustments through the canvas keyboard handler; whether they work depends on canvas keyboard focus. Zooming does not change DFLI pixels. The upper label on the graphical slider may differ from the actual software limit of 975%.

Fifth practical test — the Cut discrepancy: Edit → Cut in the menu combines copying the selected image area with preparing to fill its original location. The current source handler for `Ctrl+X`, however, only opens the Fill dialog, **without a separate Copy operation**. If you need to paste the cut-out content later, use the Cut command in the menu. Chapter 12 (Edit) provides the full explanation; this warning does not replace it.

Sixth practical test — F1, F2–F6, and F9: `F1` opens the **Shortcuts** tab of the Help / About window; `F2` – `F6` recall the previously configured Color Preset 1–5 painting-color sets; and `F9` creates a quick PNG snapshot. These serve **three different purposes**: recalling a Color Preset does not save the `.mdfli` project, and an F9 PNG is not a substitute for project saving.

Quick-reference table

Key / gesture	Action	Notes
`F1`	Help / About — Shortcuts tab	Opens the keyboard-shortcut reference tab
`F2` … `F6`	Color Presets 1–5	Applies saved color presets
`F9`	Quick Snapshot PNG	Not Blueprint
`Ctrl+Q`	Rectangle Selection	Rectangular selection
`Ctrl+F`	Freehand Selection	Freehand selection
`Ctrl+S`	Smart Area Selection	**Not Project Save!**
`Ctrl+D`	Deselect	Clears the selection
`Ctrl+C`	Copy	Copies the selection
`Ctrl+X`	Initiate Fill Color	**Does not copy automatically!**
`Ctrl+V`	Paste	Paste placement mode
`Ctrl+E`	Edit Clipboard	Clipboard editor
`Ctrl+B`	Bucket	Bucket tool
`Ctrl+M`, `Ctrl+Shift+M`	Brush Settings	Brush settings
`Ctrl+Z`, `Ctrl+U`	Undo	Undo an operation

Key / gesture	Action	Notes
`Ctrl+Y`, `Ctrl+R`, `Ctrl+Shift+Z`	Redo	Reapply an operation
`D`	DFLI grid	40×100
`G`	Pixel grid	160×200
`Caps Lock`	Character grid	40×25
`H`	Heatmap	Displays channel assignments
Hold `0` … `3`	Temporary channel override	Depends on the active tool
Hold `P`	Vertical line lock	While painting
Hold `L`	Horizontal line lock	While painting
`Page Up/Down`	Zoom	When handled by the Canvas
Numpad `+` / `-`	Change Brush Size	The main keyboard's `+/-` keys are not guaranteed
`ESC`	Cancel the active operation/preview	Context-dependent; does not undo work already committed
Right-click on Canvas	Cycle the channel code across the brush mask	Only in supported Brush modes; may have another meaning in Draw; see Chapter 6 (Brush).
Left-click on palette	Current painting color	Does not directly change image data
Right-click on palette	PlusColor FF16 painting color	Independent setting
`Alt` + left-click on Canvas	Eyedropper	Samples color and channel
`Ctrl` + left-click on Canvas	4×2 Recolor	Context-dependent
Mouse wheel	Zoom	Over the Canvas
`Shift` + Paste	4×2 alignment	During pasting

Keyboard-dependent behavior: some commands depend on Canvas focus and the active tool. The package-specific behavior of modifier keys on macOS must be checked before release; this list documents the `Ctrl` bindings visible in the examined main Java source.

24. Undo, Redo, Cancelling Operations, and Data Safety

24.1 What Do Undo and Redo Restore?

Undo restores an earlier image-data state. **Redo** reapplies a change that was undone, provided that the Redo history is still available. The current program maintains up to **64 Undo and 64 Redo** image snapshots. Both operations are available under `Tools → Edit → HISTORY`. The Undo button in the main Tools area also displays the number of available steps (for example, `Undo 07 / 64`).

The history primarily covers **DFLI image data**: the 2-bit channel codes of the 160×200 pixels, the per-raster-line FF15/FF16 colors, and the AUX color data for the 4×2 blocks. It is **not** a snapshot of the entire user workspace: an open dialog's state, the mouse position, zoom, or other display settings do not necessarily return to their previous values.

Example: Draw two lines; perform two Undo operations, followed by one Redo operation. Check which state of the image has been restored. Undo is no substitute for saving the project regularly.

24.2 Keyboard Shortcuts and HISTORY Controls

Operation	Access	Condition / result
Undo	Main Tools → Undo; Edit → HISTORY → Undo; `Ctrl+Z` or `Ctrl+U`	Steps backward if an Undo state is available.
Redo	Edit → HISTORY → Redo; `Ctrl+Y`, `Ctrl+R`, or `Ctrl+Shift+Z`	Steps forward if a Redo state is available.
Undo/Redo counter	A display such as `Undo 00 / 64`, `Redo 00 / 64`	Shows the number of available history states.

Important: After a new, actual image modification, the previous Redo chain may be cleared. Not every tool creates a separate Undo entry merely because its options were opened or Apply was pressed without making a change. Several modal tools compare the image state **before and after the operation**; such no-change operations do not consume another Undo step.

24.3 Cancel, ESC, and Close — They Are Not the Same

Control	General behavior	Important limitation
Cancel	Discards an operation that has not yet been applied in the dialog.	Does not automatically undo an image change that was already committed.
ESC	May interrupt the active drawing, selection, or paste mode.	Depends on the tool and keyboard focus; it is not a general-purpose Undo.
Close	Closes the relevant window.	Previously completed `Draw Copy` instances remain on the image.
Undo	Restores an earlier image state.	Only works while the corresponding history state is still available.

Some settings dialogs may retain certain parameter values after closing even when the image itself has not changed. **Paste Collisions** → **Cancel** may also return you to the Paste positioning mode; Chapter **12** explains Paste in detail. In every case, follow the controls and warnings of the specific tool.

24.4 Checking History and Saving Projects

Before restoring Undo/Redo data, the program checks the internal image snapshot's **expected size** and validates that its **2-bit pixel codes fall within the permitted 0–3 range**. An invalid snapshot is not restored as active image data. This is an error-protection measure, **not** a comprehensive guarantee against every kind of file corruption, disk failure, or interrupted loading operation.

The native `.mdfli` **v7** project format can preserve supported editing states in addition to the image, including up to **64 Undo + 64 Redo** history states. Chapter **18** describes project saving and loading in detail. **Quick Snapshot PNG** (`F9`), by contrast, is a quick save of the visible image for inspection; it is **not** an editable project save and does not replace an `.mdfli` file.

Recommended workflow: Before major edits or imports, save a separate `.mdfli` copy; use a new filename or version number at significant milestones; and, when useful, create a Quick Snapshot PNG of the finished image. Before exporting, inspect the current canvas, especially after Smart Preserve, Paste, or Color Filter operations.

24.5 Quick Troubleshooting

- **Undo is inactive:** No undoable state is available, or the history has been cleared, for example after starting a new project or importing an image.
- **Redo has disappeared:** You made a new image modification after Undo, which may have broken the previous Redo chain.

- **A previous drawing remains after Cancel:** The modification was already committed; try Undo if its history state is still available.
- **You cannot find an editable project after pressing F9:** F9 creates a PNG snapshot, not an `.mdfli` project save. Use Project I/O instead.

Limitation: Image-history protection does not provide transactional safety for the entire application or for every external file operation. Always keep a separate project backup for important work.

25. Anti-Aliasing — Smoothing Edges at Low Resolution

Anti-Aliasing: What Is It, and Why Is It So Important at Low Resolution?

With **Anti-Aliasing (AA, edge smoothing)**, the edge of a diagonal or curved line is not represented exclusively by pixels that are fully on or fully off. The system can insert **intermediate colors or brightness levels** between the shape and its background to make the line appear smoother to the human eye. A conventional diagonal drawn using only two sharply contrasting colors can look noticeably "stair-stepped" — this effect is called **aliasing**.

Imagine a diagonal outline:

- **Hard Pixel:** The line has a crisp, high-contrast edge with visible steps one pixel at a time. Excellent for sprite-like, iconic, deliberately pixelated styles.
- **Anti-Aliased:** Subtler shades may appear alongside the steps, reducing visual discontinuities. Useful for portraits, rounded shapes, and organic outlines.

AA **does not increase the actual resolution**. Instead, it uses the colors and shades of the existing pixels to improve the *perceived* quality of edges. Typical applications include **facial outlines, eyebrows, the curves of the nose and chin, hair, typography, circles, ellipses, car bodywork, and glowing tubes**. In retro demo graphics, combining sharply defined pixel blocks with subtly smoothed outlines can be particularly effective.

DFLI-specific challenge: In a modern 24-bit image, arbitrary intermediate RGB shades can be blended. In a Plus/4 DFLI image, a visible AA pixel must still fit within the **four available or assignable channels**, subject to the per-raster-line limits of Background FF15/PlusColor FF16 and the per-block limits of the AUX colors. AA can therefore introduce an *additional demand for colors*, potentially causing color clash. Smart Preserve may attempt to reassign channels, but the appearance of a new AA shade cannot be guaranteed in every situation. **Choosing between Hard Pixel and AA is therefore both an artistic and a hardware-related compromise.**

Practical example: On the jawline of a portrait, start with a 1–2-pixel Hard Pixel curve. If it looks too jagged, try the same curve in Anti-Aliased mode. Inspect the original shades next to the face at 400–800% zoom and again at 100% viewing size. **When comparing the results, inspect not only the edge but also the entire affected DFLI raster line or block.**

Comparison with other software: In *CorelDRAW* and many vector-graphics applications, curves and letters appear smoothed on screen; AA is introduced when the vectors are rasterized into pixels. Brawscope, by contrast, draws **directly into a 2-bit DFLI output format with strict color limitations**. The visual objective is similar, but the technical conditions differ; this is not the same vector-rendering system.

Hard Pixel and Anti-Aliased — Comparing a Specific Edge

At a logical resolution of 160×200, a diagonal line at approximately 30 degrees can only be constructed from positions on the horizontal and vertical pixel grid. **Hard Pixel** retains the silhouette's completely sharp, easily

controlled pixel boundary. **Anti-Aliased** attempts to make the curve appear less jagged by introducing additional gradations: the brightness of pixels near the edge creates a visual transition between the line and the background.

When is Hard Pixel better? Pixel-art icons, grid-aligned machine-style lettering, and high-contrast demo logos. **When is AA better?** Large curved letters, planetary edges, portrait jawlines, spotlights, and futuristic objects. In DFLI, AA does not store transparency; instead, it produces a final palette-color and channel assignment, potentially placing greater pressure on the available colors and increasing the risk of color clash. **Try this:** Use the same Line or Circle tool to draw two identical shapes, one with Hard Pixel edges and one with AA edges, and compare them at normal display size as well.

26. Glossary — Key Terms Used in This Manual

The terms below follow the meanings used in the **MeDFLIstvan v2.64 Brawscope** user interface and this manual. Refer to the indicated chapters for detailed explanations.

Term	Meaning in Brawscope	Details
DFLI (Double Line FLI)	A Commodore Plus/4 graphics display technique that uses color-map organization updated every two raster lines.	Ch. 3
Logical pixel	A single pixel in the editable 160×200 image; it is displayed with a 2:1 horizontal pixel aspect ratio.	Chs. 3, 21
2-bit pixel code / channel	A pixel's code—0, 1, 2, or 3—selects **where its visible color comes from**; it does not store the palette color index itself.	Chs. 3, 20
PaintChannel	The color carrier selected for editing: **0 Background FF15**, **1 Color1**, **2 Color2**, or **3 PlusColor FF16**. The painting color is a separate setting.	Ch. 20
Background FF15	A background color shared across a raster line; used by pixels with code 0.	Chs. 3, 20
PlusColor FF16	An additional color shared across a raster line; used by pixels with code 3.	Chs. 3, 20
Color1 / AUX1	A color shared within a 4×2-logical-pixel DFLI block; used by pixels with code 1.	Chs. 3, 17
Color2 / AUX0	A color shared within a 4×2-logical-pixel DFLI block; used by pixels with code 2.	Chs. 3, 17
TED palette index	An identifier for a color in the Plus/4 video controller's available palette; it is not the same as a pixel's 2-bit channel code.	Chs. 3, 20
Color clash	An unintended color change caused when modifying a shared FF15/FF16/AUX color carrier also affects other, non-targeted pixels.	Chs. 3, 22
Smart Preserve	A hardware-aware editing policy that, for supported operations, attempts to preserve the visible colors outside the target area; it cannot guarantee lossless results in every case.	Ch. 22
Origi Hardware	A more direct, traditional hardware-channel/register handling policy that may cause unintended color changes outside the target area.	Ch. 22
Reassign	Moving a pixel to another 2-bit channel that carries the appropriate color while keeping its visible color unchanged.	Chs. 12, 22
Collision / Safe Mode	A collision warning and a more cautious application method that may apply only part of the requested change or omit some pixels, according to the rules of the specific tool. The exact behavior is tool-dependent.	Chs. 9, 12, 22

Term	Meaning in Brawscope	Details
Selection	A selected image area created using Rectangle, Freehand, or Smart Area. It is not a universal clipping mask for every tool.	Ch. 12
Stamp / Clipboard	Image and channel information copied into the internal clipboard, which can later be inserted using Paste.	Ch. 12
Blueprint	A reference image displayed as an editing aid; it is not part of the DFLI bitmap.	Ch. 11
Palette Preserve Lift	Preprocessing controls for the Converter's eight TED luminance bands; this is not a brush-based brightening operation on the main canvas.	Ch. 19
Undo snapshot	A previous image-data state stored in the Undo/Redo history; it is not a saved copy of the entire user interface.	Ch. 24
Quick Snapshot PNG	An image file created with F9 to capture the visible result; it is not an editable `.mdfli` project.	Chs. 18, 24
Anti-Aliasing (AA)	Edge smoothing that attempts to reduce the stair-stepped appearance of pixelated outlines using available TED colors.	Ch. 25

Important distinction: PaintChannel identifies the color carrier; the **painting color** is the desired TED palette value; and the **visible image color** is determined by the pixel code together with the local FF15/FF16/AUX values. These concepts must not be confused.

27. Known Limitations and Safe Usage

This chapter summarizes the limitations **relevant to users** that are documented in the current manual. Internal patch identifiers, measured execution times on specific machines, and development test plans are not part of the user guide. The precautions described here do not replace regular project saves.

Area	Limitation or important consideration	Recommended action
Shared DFLI color carriers	Changing an FF15/FF16 raster-line color or a 4×2 AUX color can produce visible changes outside the target area.	After each operation, inspect the full affected raster lines and neighboring blocks (Chs. 3, 22).
Smart Preserve	The number of hardware color slots is limited; it is not always possible to preserve every requested color and every non-targeted pixel exactly at the same time.	Watch for Collision feedback; use Safe/Cancel, another color, or another channel when needed (Ch. 22).
Safe modes	Some of the requested changes may be omitted, and different tools may handle collisions differently.	After Apply, inspect the actual canvas rather than relying on the preview alone (Chs. 9, 12, 22).
Selection	A selection is not a mandatory clipping mask for every Brush/Draw operation; the effects of shared registers may extend beyond it.	Also inspect the area outside the selection (Chs. 12, 22).
Anti-Aliasing	Edge smoothing may require additional intermediate colors, potentially causing further color clashes.	Test it on a separate project copy first; use Hard Pixel if necessary (Ch. 25).
Color Filters	Basic/Advanced effects transform palette colors or bitmap/AUX data; they are not freely blended 24-bit RGB layers.	Preview → check Line Range → Apply → inspect the final canvas (Ch. 14).
Converter	The source preview and the image actually converted to DFLI may differ; hardware color limits take effect during conversion.	Inspect the image applied to the editor and save a separate project (Ch. 19).
Undo / Redo	Up to 64 Undo and 64 Redo image-history states; these do not automatically restore the entire UI or every file operation.	Save a separate `.mdfli` project copy before major steps (Ch. 24).

Area	Limitation or important consideration	Recommended action
Project Save / Load	This manual does not promise complete atomic recovery for every corrupted file, interrupted save, or failed load.	Do not overwrite your only original project file; maintain versioned backups (Chs. 18, 24).
PRG Import	Importing may establish a new image-data state, so unsaved work in the current edit may be lost.	Save a separate project before importing (Ch. 18).
Ctrl+X	This shortcut behaves differently from Edit → Cut: it does not perform an automatic Copy operation.	For reliable cutting, use Edit → Cut or Ctrl+C followed by a separate fill operation (Ch. 23).
Zoom	The slider is labeled up to 1000%, but the current canvas's actual maximum zoom is 975%.	Check the actual value in the Status bar (Ch. 21).
Kaleidoscope	Symmetry is applied only through supported Brush/Draw paths; other operations do not necessarily produce symmetrical copies.	Check tool support and start with a short test stroke (Ch. 10).
Platforms	Display behavior and file chooser dialogs may differ across Windows, Linux, and macOS environments.	Verify dialogs and save locations on your own installation (Chs. 2, 18).

27.1 Minimum Data-Safety Precautions

Before a major transformation, save a separate `.mdfli` project. Create a new version before significant imports, Paste operations, Color Filter operations, Converter operations, or larger Smart Preserve edits. An **F9 Quick Snapshot PNG** is useful for comparing the visual result, but it **does not** preserve the project's editable DFLI state. See **Chapters 18 and 24** for details.

27.2 What This Manual Does Not Promise

Brawscope operates within the constraints of the Plus/4 hardware; neither Smart Preserve, Anti-Aliasing, nor the Converter guarantees a perfect reproduction of an arbitrary RGB target image. With some tools, the **Preview** shows the intended effect rather than every consequence of the final hardware collision handling. The definitive result is the **applied DFLI image displayed in the editor**.

28. Combined Example Project — Using Multiple Tools Together

This chapter presents a **tutorial workflow**, not a separate menu command. The goal is to create a Commodore Plus/4 demoscene-style title screen featuring a raster-bar background, a symmetrical ornament, and a geometric emblem. The detailed controls of individual features are not repeated here; they are covered in the referenced chapters.

28.1 Workflow Overview

Step	Tool	Expected Result	Details
1.	Project I/O	A safe, editable starting project	18, 24
2.	Magic Raster Tool	Horizontal bands using FF15 or FF16 colors	16
3.	Live Kaleidoscope / Radial	Supported Brush strokes repeated around a center point	10
4.	Draw / Circle or Star	A regular central shape	7, 25
5.	Redraw Shape / Draw Copy	Repositioned, rotated, or scaled copies	7
6.	Color Presets	Two different color sets for subsequent painting operations	15
7.	Project save and export	Editable `.mdfli` project and PRG/ASM/PNG output	18

28.2 Preparation and Checkpoints

1. Back up your work. Start with a new project or one saved under a different name. Save an initial `.mdfli` project state. **F9** saves a PNG image, but it does not replace the native project file (Chapters 18 and 24).

2. Working view. Set an appropriate Zoom level for editing and, if needed, press **D** to enable the 40×100 DFLI block grid. This grid is a visual editing aid, not part of the exported bitmap (Chapter 21).

3. Areas to inspect. Examine the outline of the central shape separately from the points where the horizontal color bands intersect it. Geometry, AA, and 4×2 block colors can affect the former; shared, per-raster-line FF15/FF16 values can also affect the latter (Chapters 20–22 and 25).

28.3 Magic Raster — Horizontal Color Bands

Open the **Magic Raster Tool** window. Select **Background FF15** or **PlusColor FF16** as the target channel, specify a relatively short **Start–End** raster-line range (for example, 50–90), and choose a factory **Preset**. Examine the color-strip preview: it displays the colors assigned to individual raster lines, **not a final preview of the entire DFLI canvas**. After clicking **Apply**, check the full width of the affected raster lines. If you want to create your own band progression, see Chapter **16** for the detailed description of the **8×25 Pattern Editor**.

Expected appearance: a sequence of horizontal raster bars with varying colors, visible in pixels that reference the corresponding FF15/FF16 channel on the affected lines. **Limitation:** this operation does not redraw the 2-bit bitmap codes.

28.4 Kaleidoscope — Radial Ornamentation

Activate a **Live Kaleidoscope** session. Select **Radial** in the **Style** field and start with **Area: Free Drawing**. Position the center near the intended emblem, select a supported **Solid Brush** setting, and draw a short test stroke. If the rotated copies appear in the desired positions, continue building the motif. For comparison, return to the same saved starting state and try the **Kaleidoscope** Style, which also applies mirroring.

Green Sector Only controls the permitted source-drawing area; it is **not** another symmetry style. The two selectors are independent (Chapter 10). **Limitation:** numerous symmetrical copies can affect more DFLI blocks and raster lines, so any resulting color clashes must be checked.

28.5 Draw and Redraw Shape — Emblem and Copies

Use **Draw** → **Circle** or **Draw** → **Star** to create a simple central shape with a clearly defined outline. Begin with **Hard Pixel** edges; try the **Anti-Aliased** version in a separate copy of the project. AA aims to produce visually smoother edges, but it may require additional intermediate colors that DFLI limitations do not always permit (Chapters 7 and 25).

You can duplicate a successfully completed, supported shape using **Redraw Shape** → **Draw Copy**. For **Box, Circle, Triangle, and Star**, **Scale 10–800%** is also available; combine it with X/Y and Rotation settings to create different arrangements. **Close** does not undo copies that have already been finalized. See Chapter **7** for details.

Check: inspect the visible colors of pixels outside the intended targets around the emblem and its copies as well. The additional checks required depend on the **Smart Preserve / Origi Hardware** selection (Chapter 22).

28.6 Color Presets — Two Painting Color Schemes

Open the **Color Presets** window (Chapter 15). In one row, use **Capture Current** to capture the four currently selected painting colors; in another row, use the color cells to configure a warmer set. Enter names for the presets, then use **F2–F6** after clicking **Close** to switch among the five presets. You can also preserve the set in a separate `.medpreset` file using **Save Presets**.

Important: switching presets changes the colors selected **for subsequent painting operations**; it does not recolor the entire existing DFLI image. To compare the results of the two color schemes, perform actual new drawing operations in separate project copies.

28.7 Final Image Inspection and Export

Turn off the guide grids, inspect the picture at normal size and at higher zoom levels, and use **Pointer** and **Heatmap** to examine critical areas (Chapter 21). Do not inspect only the intended shape; also check any entire raster lines and 4×2 blocks that may have been affected (Chapter 22).

First save a separate, editable `.mdfli` **project**, then perform a **PRG + ASM + PNG** export if required (Chapter 18). The resulting PNG is useful for viewing the picture, whereas the native project is needed for continued editing. If an operation produced an uncertain result, return to a previous project copy or use **Undo** while the relevant history entry remains available (Chapter 24).

29. License and Distribution

The program is an **independent fan development** and was not produced with the approval or support of Commodore. The current About text identifies Commodore Plus/4 as a trademark of Commodore Business Machines.

The software is **freeware** and may be used and distributed **free of charge and in unmodified form**. Commercial redistribution, sale, repackaging, or inclusion in a paid collection is prohibited without the author's prior written permission.

The software is provided **“as is”**, without any warranty of any kind, and is used at your own risk.
Copyright © 2026 - ByteHexer / Monarchy. All rights reserved.

Endnotes

Special thanks to Istvanv

Special thanks to Istvanv for allowing MeDFLIstvan to use his Plus/4 DFLI PRG viewer! It was a deliberate decision **not to create yet another competing DFLI file format**. We believe it is far more important for Plus/4 DFLI images to remain, wherever possible, in a consistent, practical format compatible with other applications. Istvanv's solution provides an excellent foundation: it is well thought out, versatile, and, we believe, still has plenty of untapped potential.

The purpose of the chosen PRG format is **not to pursue the absolute highest image quality at any cost**, but to provide **versatility and practical usefulness**: graphics should be easy to incorporate into demos, games, and other Plus/4 projects. Naturally, the Brawscope editing and conversion tools still aim for the best possible visual result, while preserving the advantages of an established, shared DFLI output format.

We also want to promote **Istvanv's work and the p4fliconv application**, the **MeDFLIstvan / Brawscope** editor, and above all **the idea of a unified Plus/4 DFLI format**. We are not starting another format war; we are building together on something that already works. **One shared format, two programs, countless possibilities: everything is free, and the Plus/4 community benefits from both. Promoting each other means promoting ourselves... Wait, what!? Yes, really!** :) :)