ZX Touch

FX Guide

Transforming ZX Spectrum Graphics Without Modifying the Original Game

Table of Contents

1. Introduction

ZX Touch FX is a visual enhancement system for original ZX Spectrum games. It allows the appearance of a game to be dramatically improved by using colour palette changes, edge shading, transparency, background images and automatic scene-based switching through FX Manager.

A key point is that **the original game is not modified**.

FX works as a **dynamic post-processing layer** applied to the image produced by the emulator. The ROM, game logic, timing and gameplay remain exactly the same as in the original game. Even when the visual difference is very large, the game is still running from the original unmodified ROM.

This is why some FX-enhanced games may look almost like remakes, even though they are still the original ZX Spectrum games.

This document is written primarily for creators and advanced users who want to build their own FX-enhanced games. It explains how the FX system works, how its parts interact, and how to use them effectively when preparing a ZTG package.

At the same time, many users will not create their own FX configurations, but will simply use ready-made FX game packages prepared by others. For such users, only one preparation step is required.

For users who only want to play ready-made FX games

Ready-made ZTG packages distributed from official sources do not include the original game ROM. To use such a package, you must add your own legally obtained original game file.

To do this:

1. Copy the ZTG file and the original game file into the same folder on the SD card.
2. Rename the original game file so that it has exactly the same name as the ZTG file, with only the extension being different.
3. In the file browser, press and hold the ZTG file.
4. Select **Edit**.

If the option **Include the game in the ZTG file** is already checked, this means the ZTG file already contains the game and no further action is required.

5. Enable the option **Include the game in the ZTG file**.
6. Press **Finish**.

The original game is now embedded into the ZTG file together with all settings.

If you want to add it to the normal ZX Touch interface:

7. Press and hold the same ZTG file again.
8. Select **Add to Game Library**.

The game can then be added from the Library to any dashboard.

For users who only want to play FX-enhanced games, that is all that is required.

The rest of this document is intended for users who want to create, edit and fine-tune FX configurations themselves.

2. FX System Overview

The FX system in ZX Touch enhances the visual appearance of ZX Spectrum games by applying real-time processing to the image generated by the emulator.

The system consists of five main components:

- **Colour Palette** defines the base colours
- **Edge Colour Shader** enhances edges and modifies sprite appearance
- **Foreground Transparency** controls visibility of the game image
- **Backgrounds** provide external imagery behind the game
- **FX Manager** controls how and when FX configurations change during gameplay

Together, these components form a layered system where the original ZX Spectrum image is processed step by step into the final enhanced visual output.

Each component can be used independently, but the most effective results are achieved when they are combined.

Additional tools that support FX creation:

- Screen Capture (section 3.7)
- FTP server (for fast file transfer between the console and a PC) (section 4.1a)

2.1 How FX Works

The emulator first renders a standard ZX Spectrum frame using pixel and attribute data. The FX system then processes this image in real time.

Processing order: The five components are applied in a fixed sequence on every frame. First, the Colour Palette transforms the base ZX colours. Second, the Edge Colour Shader analyses pixel neighbourhoods and assigns L1/L2 edge classifications. Third, Foreground Transparency blends the resulting image with the background. Fourth, In-game Background images are revealed behind any pixels whose colour was selected as transparent — except edge pixels, which remain visible as outlines. Fifth, if FX Manager is active, it monitors the current frame for markers and may instantly switch all of the above parameters to a different slot. Understanding this order is important: a change to the palette affects how the shader calculates edges, and a change to the shader affects which pixels become transparent.

This processing can include:

- modifying colours through the Colour Palette
- enhancing edges and modifying sprite appearance using the Edge Colour Shader
- applying transparency to selected colours
- blending background images behind the game
- adjusting overall foreground visibility

All FX operations are applied per frame and can change dynamically during gameplay, while the original game continues to run unchanged.

The same game can be transformed in many different ways depending on the chosen palette, shader configuration and background design. This means there is no single correct result, but a wide range of possible visual styles.

2.2 FX Manager Concept

FX Manager is responsible for dynamic switching of FX configurations during gameplay.

Instead of using a single visual setup for the entire game, FX Manager allows different configurations to be applied depending on what appears on the screen.

This is based on a **marker → action** principle:

- A **marker** is a small, predefined fragment of the screen
- When the system detects that marker, it triggers a corresponding **action**

Each action defines:

- which **background image** will be used
- which **transparent colour (or colours)** will be applied
- which **FX slot** will be activated

An FX slot contains a complete configuration of:

- Colour Palette
- Edge Colour Shader
- Foreground Transparency

When a marker is detected, FX Manager instantly switches to the assigned background and FX slot.

What is an FX slot? An FX slot is a named storage container that holds a complete visual configuration — Colour Palette, Edge Colour Shader settings, and Foreground Transparency — all saved together as a single unit. FX Manager provides 10 slots (0–9). When an action is triggered, the system instantly loads the assigned slot into the active display parameters. This is what makes instant scene switching possible: you configure each visual style once, save it into a slot, and the system swaps between them automatically during gameplay.

2.3 Quick Workflow Overview

Before reading the detailed reference sections, it helps to understand the complete FX creation process from start to finish. This overview describes the typical path from a raw game file to a fully enhanced FX package.

Step 1 — Create a base ZTG file. Place the game and at least one background image into a ZTG file. Set up key mappings and any console preferences required for comfortable gameplay.

Step 2 — Build the base visual style. With FX Manager disabled, work on Colour Palette, Edge Colour Shader, Foreground Transparency, and the first background image. Select the transparent colour so the background shows through the game. Adjust the shader so objects have clear outlines. Iterate until you have a consistent, readable look for the main gameplay screen.

Step 3 — Use PNG screenshots to design backgrounds. Take a PNG screenshot from the console. The captured image includes your current palette and shader output. Open it in a PC image editor, use it as a reference layer, and paint or composite your background behind it. This ensures perfect alignment between background art and game graphics.

Step 4 — Store configurations into FX slots. Once you are satisfied with a visual setup, open FX Manager, select a slot number, and press Store to Slot. Repeat this for each distinct visual style the game needs (for example, one slot for the menu, one for gameplay, one for each level type). You can have up to 10 slots.

Step 5 — Capture markers and link them to actions. Pause the game at a screen you want to detect (for example, the title screen, or the point where level 1 starts). Use the Game screen cursor to select a small stable area on screen — something like a score display, level number, or static logo. Capture it as a marker. Then define the action for that marker: which background image to show, which transparent colour to apply, and which FX slot to activate. Repeat for each scene transition.

Step 6 — Test and iterate in RAM. Switch to a RAM-based workflow: prepare a ZTG file without stored parameters so that starting the game does not overwrite your current configuration. Play through the game, observe the transitions, return to Settings and adjust. Use Save Config and Load Config to preserve your work between sessions without needing to update the ZTG file every time.

Step 7 — Finalise and distribute. When everything works correctly, enable parameter saving and update the ZTG file. Restart the game from the ZTG file to verify clean behaviour. The ZTG file now contains all FX data and is ready for distribution. If distributing to others, you can exclude the game code and let recipients add their own legally obtained copy.

The sections that follow explain each component in detail. They can be read in order or used as reference when working on a specific part of the configuration.

3. FX Options in Display Settings

All FX-related options are located together in the **Display settings** menu of the console.

Figure 1. FX options in Display settings

From this menu, each FX component can be enabled, disabled, and configured independently:

- Colour Palette
- Edge Colour Shader
- Foreground Transparency
- In-game Backgrounds
- FX Manager

Each of these options includes an **Edit** button that opens its configuration screen.

Although the FX components work together, each one can also be adjusted separately. This makes it possible to start with simple visual improvements and later build more advanced configurations.

3.1 Colour Palette

The Colour Palette defines the base colours used to display ZX Spectrum graphics.

It is one of the most important parts of the FX system because it changes the overall visual character of the game and also affects how other FX components behave, especially the Edge Colour Shader.

The Colour Palette can be used in two ways:

- by selecting one of the predefined presets
- by creating a fully customised palette

3.1.1 Colour Palette screen

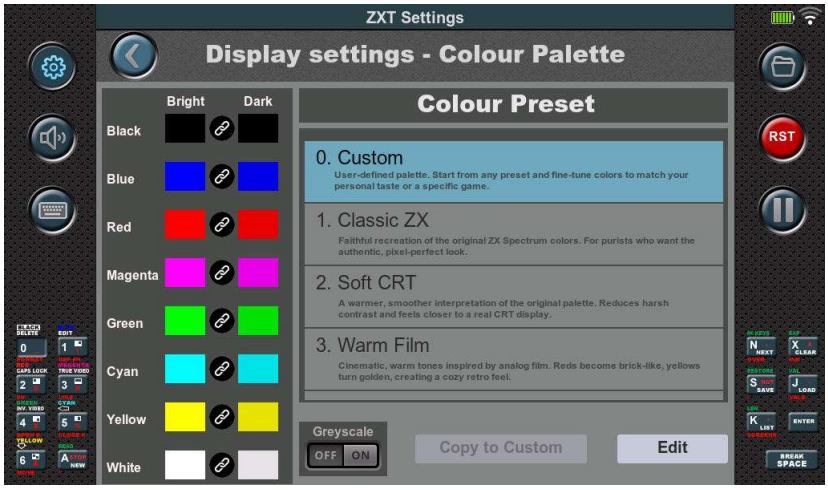


Figure 2. Colour Palette – preset selection screen

When the Colour Palette screen is opened, the left side shows the current palette arranged in ZX Spectrum colour order.

For each of the 8 ZX Spectrum colours, two variants are shown:

- **Bright**
- **Dark**

This gives a total of 16 displayed colours.

The right side of the screen contains the list of available **Colour Presets**.

At the bottom are the following controls:

- **Greyscale**
- **Copy to Custom**
- **Edit**

3.1.2 Colour Presets

The Colour Palette includes a set of predefined presets that instantly change the appearance of the game.

These presets are:

- **0. Custom**
- **1. Classic ZX**
- **2. Soft CRT**
- **3. Warm Film**
- **4. Cool Glass**
- **5. Desert Sun**
- **6. Ocean Depth**
- **7. Forest Night**
- **8. Vaporwave**
- **9. C64**
- **10. Green Phosphor**
- **11. Amber Phosphor**

Each preset defines a complete palette configuration.

Some presets are designed to remain close to the original ZX Spectrum look, while others provide a more stylised appearance.

For example:

- **Classic ZX** keeps the original colour spirit
- **Soft CRT** gives a softer and less aggressive look
- **Warm Film** shifts the image toward warmer tones
- **C64**, **Green Phosphor** and **Amber Phosphor** deliberately evoke other display styles

3.1.3 Custom preset

The **Custom** preset is special.

It is the only preset that can be edited directly.

If any other preset is selected, the **Edit** button is disabled. In that case, if you want to start from that preset and then adjust it manually, you must first press:

Copy to Custom

This copies the currently selected preset into the Custom preset.

After that, select **Custom** and press **Edit**.

This workflow is important:

- presets are used as starting points
- **Custom** is the editable working preset

3.1.4 Greyscale

The **Greyscale** option converts the currently active palette into monochrome.

This is not a separate preset. It is an additional display mode applied to the currently selected palette, whether that palette is a preset or a custom configuration.

This means that Greyscale can be used together with:

- Classic ZX
- any other preset
- Custom

It can be useful for creating a black-and-white display look or for testing how a palette behaves without colour.

3.1.5 Editing the Custom palette

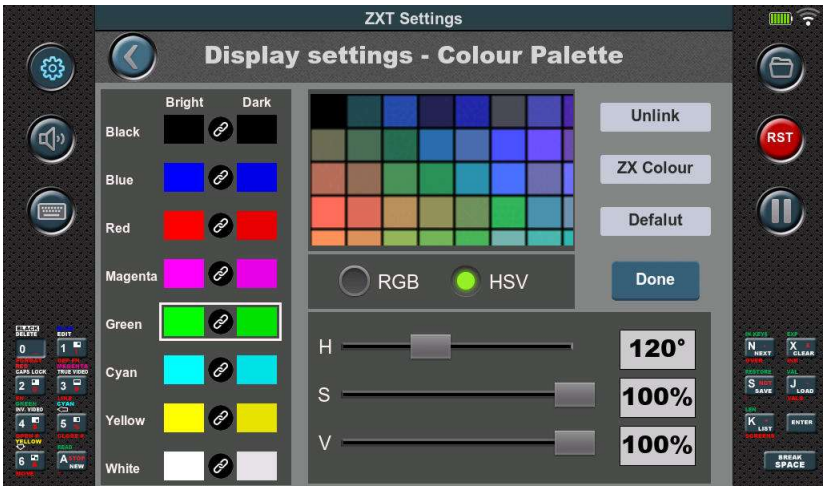


Figure 3. Colour Palette – Custom editing screen

When the **Custom** preset is selected and the **Edit** button is pressed, the palette editing screen is opened.

This screen allows direct editing of individual colours.

On the left side, all 8 ZX Spectrum colours are shown again, each with:

- **Bright**
- **Dark**

A colour pair can be selected for editing by tapping it. The selected colour is highlighted.

3.1.6 Link and Unlink

Between the **Bright** and **Dark** columns there is a link icon.

By default, Bright and Dark variants of the selected ZX colour are linked together.

When they are linked:

- editing the main colour also updates its paired variant automatically

If you press **Unlink**, the two variants become independent.

This allows you to edit:

- the Bright variant separately
- the Dark variant separately

This is useful when you want more precise control over contrast, shading and colour relationships.

3.1.7 Colour selection area

The upper middle section contains a colour grid.

This grid provides a quick way to choose a starting colour visually. After selecting a colour from the grid, it can still be fine-tuned using sliders.

This makes editing faster, because you do not always need to build a colour from zero.

3.1.8 RGB and HSV modes

Below the colour grid there are two editing modes:

- **RGB**
- **HSV**

RGB mode

RGB mode allows direct adjustment of:

- **R** – Red
- **G** – Green
- **B** – Blue

This is useful when exact channel values are needed.

HSV mode

HSV mode allows adjustment of:

- **H** – Hue
- **S** – Saturation
- **V** – Value

This is often more intuitive for visual tuning, because it allows the colour tone, intensity and brightness to be adjusted separately.

The user can switch freely between RGB and HSV while editing the same colour.

3.1.9 Sliders and numeric values

The lower part of the screen contains sliders for the currently selected mode.

On the right side of the sliders, the exact numeric values are shown.

This allows both:

- rough visual adjustment with sliders
- precise tuning using the displayed values

3.1.10 Editing buttons

On the right side of the editing screen there are four buttons:

Unlink

Separates the Bright and Dark variants of the selected colour pair so that they can be edited independently.

ZX Colour

Resets the selected colour to its original ZX Spectrum value.

This is useful when a colour needs to be restored without affecting the rest of the palette.

Default

Restores the selected colour from the saved default configuration.

This is not the same as ZX Colour.
ZX Colour returns to the original ZX Spectrum colour, while **Default** returns to the saved default state of the console configuration.

Done

Exits the editing screen and returns to the main Colour Palette screen.

3.1.11 Important behaviour

The Colour Palette system preserves all user changes.

This is especially important for the **Custom** preset.

If the user edits Custom, then later switches to another preset, the edited Custom values are not lost. When Custom is selected again, the previously edited colours are restored exactly as they were before.

This behaviour also applies when configurations are stored in FX slots.

A slot stores the full palette state, including:

- the currently selected preset
- Greyscale state
- all Custom palette data

This means that restoring a slot returns the Colour Palette screen to the same state in which it was saved.

For example, if a slot was saved while:

- **Warm Film** preset was selected
- **Greyscale** was enabled
- **Custom** already contained user-edited colours

then restoring that slot will:

- select **Warm Film**
- enable **Greyscale**
- preserve the stored Custom palette data in the background

If the user later switches back to **Custom**, those saved Custom colours will still be there.

This behaviour allows experimentation without losing previous work.

3.1.12 Practical use

A typical way to work with the Colour Palette is:

1. Select a preset that is close to the desired style
2. Press **Copy to Custom**
3. Select **Custom**
4. Press **Edit**
5. Adjust individual colours
6. Save the result into an FX slot if it will be used by FX Manager. To do this, open FX Manager, select the desired slot number, and press Store to Slot. If you are not using FX Manager, save the palette through the standard Save/Load settings menu instead.

This makes the preset system a fast starting point, while Custom provides full manual control.

Important: Because the Colour Palette is applied first in the processing pipeline, changing it also changes the base colours that the Edge Colour Shader uses for L1 and L2 calculations. Any time you modify the palette, review the shader results and adjust L1/L2 colours if needed. A shader style that looked correct with one palette may need adjustment after the palette changes.

3.2 Edge Colour Shader

The Edge Colour Shader enhances ZX Spectrum graphics by improving object separation and increasing perceived colour depth.

A key function of the shader is the creation of a controlled outline around objects. This outline plays a crucial role when backgrounds are used, as it visually separates foreground elements from the background image and prevents them from appearing artificially overlaid.

At the same time, the shader enriches the interior of objects. Due to the limitations of the original ZX Spectrum hardware, many objects are effectively single-colour. The shader introduces additional colour variation along edges, allowing these objects to appear more detailed and visually richer without altering the original game data.

These two effects work together: the outline defines the object clearly against the background, while internal edge shading adds depth and improves the overall visual impression.

Two shader modes are available. Standard mode provides the most accurate and visually complete result, while Light mode offers a simplified and faster alternative intended for performance-critical scenarios such as turbo gameplay.

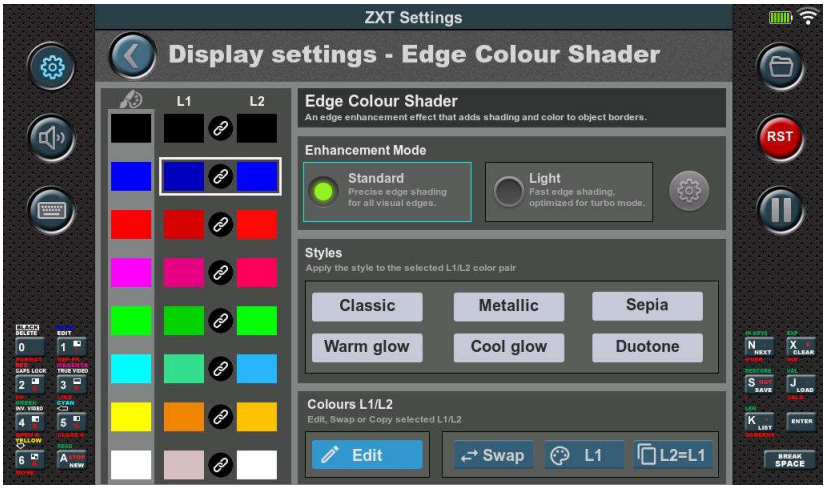


Figure 4. Edge Colour Shader – main screen

3.2.1 Basic concept

The shader analyses each pixel in relation to its immediate neighbours and detects transitions between different colours. These transitions define visual edges in the image.

Based on this analysis, edge pixels are treated differently from inner pixels. Pixels that form edges are reassigned to L1 or L2 levels, while pixels fully surrounded by the same colour remain unchanged, or become transparent if their ZX colour is selected as a background colour.

Importantly, pixels that belong to an edge will not become fully transparent, even if their original ZX colour is selected as a background colour. Instead, they are transformed into L1 or L2 colours. This behaviour creates a visible outline around objects, which is essential for proper visual integration with background images.

This processing is applied on every frame of the original game output.

3.2.2 Edge levels (L1 and L2)

Each ZX Spectrum colour has two edge levels:

- **L1 (Level 1)**
- **L2 (Level 2)**

These are not layers, but classifications of pixels based on their surroundings.

- **L1 pixels** are pixels close to the edge, surrounded by fewer pixels of the same colour
- **L2 pixels** are slightly deeper inside the object, but still influenced by neighbouring colours
- pixels fully surrounded by the same colour are not part of L1 or L2

Those inner pixels keep their base colour from the Colour Palette, or become transparent if their ZX colour is selected as the background colour.

3.2.3 Shader Modes

The Edge Colour Shader provides two operating modes that differ in precision and performance.

Standard mode delivers the most accurate result. It analyses the full image and detects edges even in flat colour areas where pixel transitions are not clearly visible. This allows consistent outline generation and more natural integration with background images.

Light mode uses a simplified approach based on pixel transitions. It is significantly faster but may miss subtle edges, especially in large flat areas. This mode is intended for performance-critical scenarios such as turbo gameplay.

In most cases, Standard mode provides the best visual quality, while Light mode offers a practical trade-off when performance is more important.

3.2.4 Shader Modes Options

Additional shader behaviour can be configured using the Options button next to the Standard / Light mode selection.

This opens a dialog with advanced processing options.

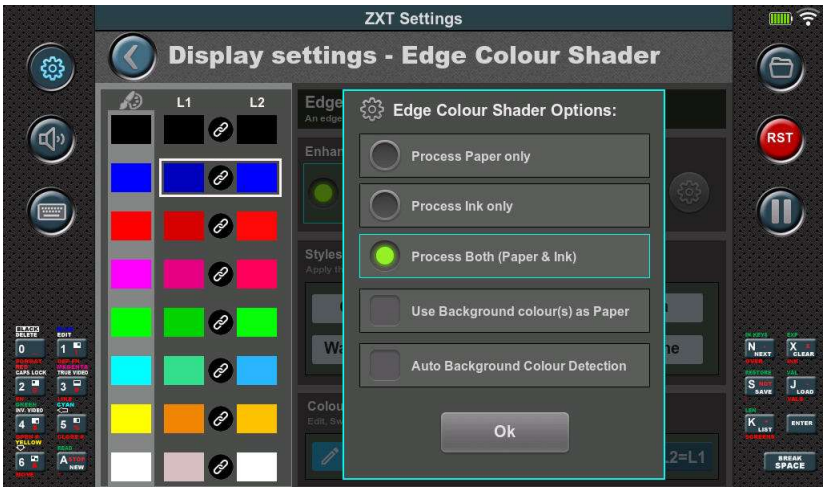


Figure 4a. Shader Modes Options

Processing scope

The shader can be limited to specific pixel types based on how ZX Spectrum graphics are stored:

- **Process Paper only** – shader is applied only to pixels that belong to paper
- **Process Ink only** – shader is applied only to pixels that belong to ink
- **Process Both (Paper & Ink)** – shader is applied to both types of pixels

In most cases, Process Both provides the most balanced and natural result. The other modes can be useful for specific visual effects or when fine control is required.

- **Use Background Colour(s) as Paper**

This option is available only when Process Paper only or Process Ink only is selected.

When enabled, any pixel whose resulting colour matches the selected background colour is treated as paper, regardless of its original ZX memory classification.

This can be useful in cases where games use non-standard drawing techniques or when background colours are reused in different roles.

When Process Both is selected, this option has no effect because both pixel types are already processed.

- **Auto Background Colour Detection**

Auto Background Colour Detection allows the shader to automatically determine which ZX colour should be treated as the background by analysing the current screen, instead of using a fixed transparent colour.

This is useful when FX Manager is not used and the game changes its background colour between scenes or levels. The shader adapts automatically, allowing a single setup to work across different screens.

When FX Manager is enabled, this option is ignored, as transparent colours are controlled per action.

3.2.5 Practical colouring rules

In practice, the following approach gives the best results:

Background side (outline):

- for the ZX Spectrum colour (or colours) that are used as transparent
- set **L1 and L2 to the same colour** for the ZX colour that is used as transparent

This defines the colour of the outline.

Example:

- if the transparent colour is black and you want a black outline → L1 = black, L2 = black

Object side (shading):

- set **L1 and L2 to similar but not identical colours**
- this creates a shading effect

This can be achieved by adjusting:

- brightness (lighter/darker)
- hue (slight colour shift)

This combination produces:

- a clean outline on the background side
- smooth shading on the object side

3.2.6 Shader Styles

The shader includes predefined **styles**.

Each style automatically assigns L1 and L2 values based on the base colour from the Colour Palette.

Available styles include:

- Classic
- Metallic
- Warm glow
- Cool glow
- Sepia
- Duotone

Important behaviour:

- styles are applied **per colour**
- the result depends on the current Colour Palette
- the same style produces different results for different base colours

3.2.7 Quick Editing Tools

Several quick editing buttons are available next to the Edit button. These tools allow fast adjustment of L1 and L2 values without entering the colour editor.

Swap

The Swap button exchanges L1 and L2 values.

This is often useful after applying a shader style, when the selected colours are suitable but assigned to the wrong levels. It provides a quick way to correct the overall look without manual editing.

Copy Palette Colour → L1

This button copies the base colour from the Colour Palette into the L1 value.

It is useful when you want L1 to match the original palette colour, for example when defining a clean outline or resetting shading.

Copy L1 → L2

This button copies the current L1 value into L2.

This is commonly used when you want both edge levels to use the same colour, for example when creating a solid outline instead of a shaded transition.

3.2.8 Editing shader colours

Pressing **Edit** opens the colour editor.

Here you can manually adjust L1 and L2 values for each ZX colour.

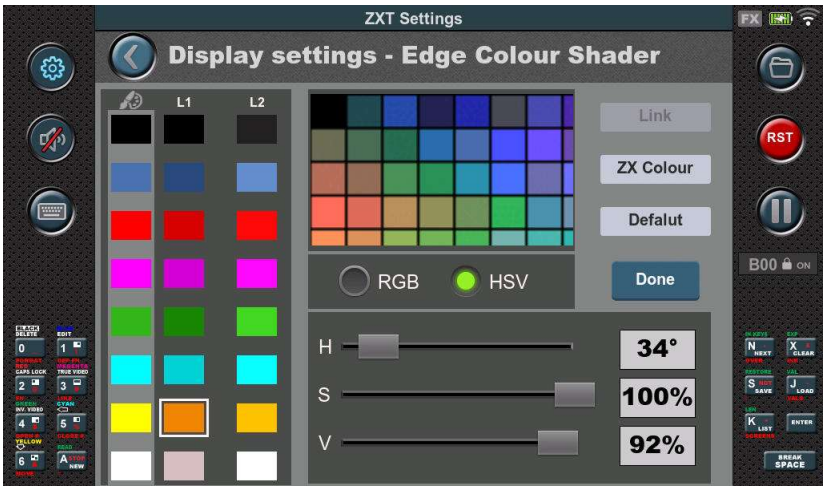


Figure 5. Edge Colour Shader – colour editing screen

Controls include:

- colour grid for quick selection
- RGB / HSV modes
- sliders for adjustment

The numeric values shown next to sliders are **display-only**:

- they cannot be entered manually
- they help track changes and compare values
- this is useful because small differences are difficult to judge visually

Buttons:

- **ZX Colour** – resets to original ZX value
- **Default** – restores default
- **Done** – exits editor

Important:

- L1 and L2 are always edited independently
- linking behaviour exists only when styles are applied

3.2.9 Important notes

- L1 and L2 are applied dynamically based on pixel neighbourhood and are not fixed to specific positions. The same pixel may belong to L1, L2 or neither depending on the current frame.
- Edge pixels are always processed by the shader and will not become fully transparent. This ensures that objects remain clearly defined when backgrounds are used.
- The final visual result depends strongly on the Colour Palette. The same shader configuration can produce very different results when used with different palette settings.
- Shader styles provide a fast starting point, but manual adjustment of L1 and L2 is often required for optimal results.
- Different FX configurations can produce equally valid results. The system does not enforce a single visual style, and the final appearance depends on the user's choices and design approach.

3.3 Foreground Transparency

Foreground Transparency controls the visibility of the ZX Spectrum image relative to the background.

It does not define which pixels are transparent — that is handled by colour selection.
Instead, it affects how strongly the remaining (non-transparent) parts of the image are blended with the background.

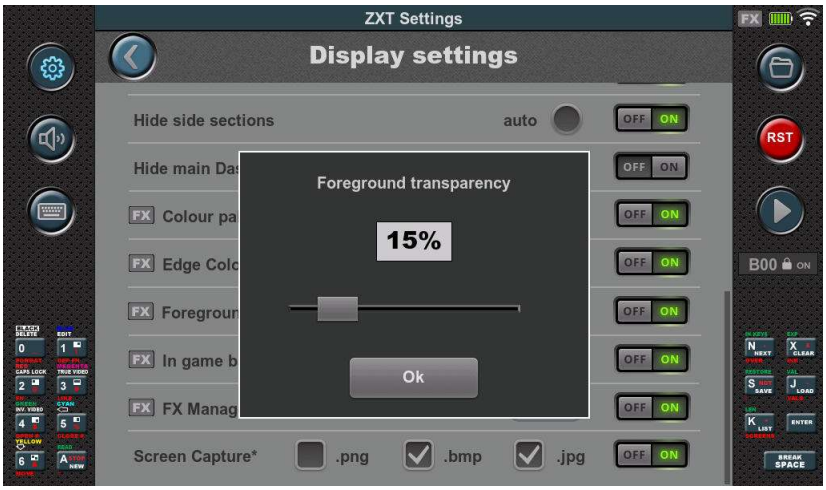


Figure 6. Foreground Transparency – adjustment screen

3.3.1 Basic concept

After colour palette adjustments, shader processing and transparency selection are applied, the remaining visible parts of the ZX Spectrum image can be partially blended with the background.

Foreground Transparency controls this blending.

- **0%** → ZX Spectrum image is fully visible
- **100%** → ZX Spectrum image is completely invisible

This is applied uniformly across all non-transparent pixels. Pixels that are already fully transparent (based on selected transparent colours) are not affected.

3.3.2 Scenario 1 – Full replacement (100%)

When Foreground Transparency is set to **100%**, the ZX Spectrum image becomes completely invisible.

Only the background remains visible.

This is useful for:

- loading screens
- static menus
- screens without animation

In these cases, the original image is effectively replaced by a high-resolution background, without colour limitations.

The game is still running normally, but its visual output is completely hidden.

3.3.3 Scenario 2 – Adding texture to solid areas

In some cases, a large solid colour area should not be made fully transparent because:

- it contains animation
- or important elements that must remain visible

Instead of removing it, Foreground Transparency can be used to enrich it visually.

The approach is:

1. Keep the solid colour visible (do not mark it as transparent)
2. Design a background with a strong, high-contrast pattern in that area (e.g. wood, metal, stone)
3. Adjust the Colour Palette so that the base colour fits the intended material
4. Set Foreground Transparency to a low value (typically **10–15%**)

Result:

- the palette-defined colour remains dominant
- the background pattern subtly shows through
- the flat colour gains texture and depth

This effect is usually barely noticeable on detailed parts of the image, but clearly visible on large uniform areas.

3.3.4 Scenario 3 – Blending foreground with background

Foreground Transparency can also be used to reduce the visual contrast between the ZX Spectrum image and the background.

If the foreground appears too strong or detached from the background, a small transparency value can help visually integrate both layers.

Typical usage:

- low values (10–25%)
- applied to soften the image

This makes the final image feel more cohesive, especially when using detailed backgrounds.

3.3.5 Important notes

- Foreground Transparency does **not** affect which pixels are transparent
- it only affects pixels that remain visible after transparency processing
- it is applied uniformly across the entire image

Small values often produce the best results, while higher values should be used only in specific cases such as full replacement.

The final effect depends strongly on:

- the Colour Palette
- the Edge Colour Shader
- the design of the background image

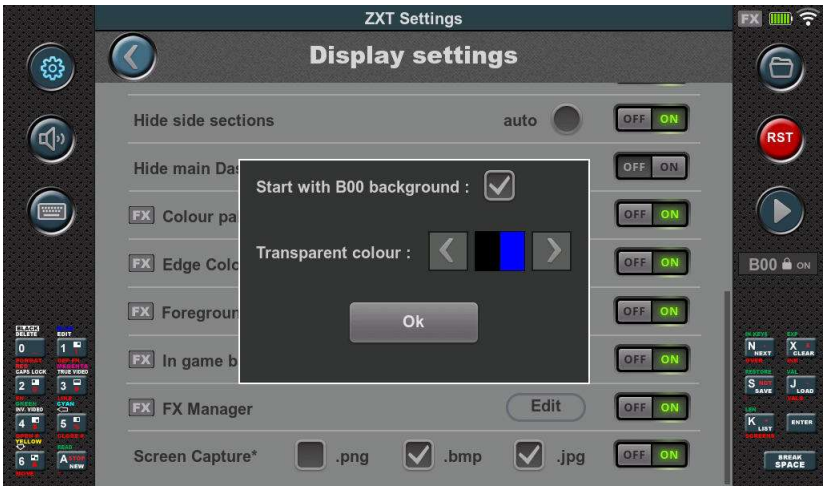
3.4 In-game Backgrounds

In-game Backgrounds allow ZX Spectrum graphics to be visually enhanced by placing a custom image behind the original game output. This is achieved by making selected ZX colours transparent, revealing a high-resolution background image underneath. This process is applied in real time on every frame of the original game output.

3.4.1 Basic Concept

The primary purpose of backgrounds is to replace large flat colour areas of the ZX Spectrum image with visually richer content. ZX Spectrum games often use solid colours to represent background surfaces such as sky, ground, or empty space. By making these colours transparent, those areas are replaced with detailed imagery. The original game graphics remain unchanged and continue to define gameplay and structure.

3.4.2 Transparent Colours



(Figure 7 – In-game Backgrounds Edit window)

The system supports selecting one or two transparent colours.

In most cases, a single transparent colour is sufficient. This is typically used when the game has a single dominant background colour.

However, two transparent colours are useful in more complex situations.

For example:

- one colour may represent the gameplay background (e.g. black)
- another colour may be used in HUD or interface elements (e.g. cyan)

If both areas should reveal the background image, two transparent colours must be used.

Another common case is when a scene contains two large regions, such as sky and ground, each using a different colour.

Only pixels using the selected ZX colours become transparent. All other pixels remain fully visible. Transparent colours are selected using left/right controls.

Two identical colours cannot be selected at the same time. When one colour is already selected, it is automatically skipped when choosing the second one.

3.4.3 Interaction with Shader and Palette

Transparency is evaluated after colour palette adjustments and shader processing.

Pixels processed by the Edge Colour Shader (L1 and L2) are no longer fully transparent, even if their original ZX colour is marked as transparent.

This behaviour is a major advantage. Because edge pixels remain visible, objects retain a clean outline when placed over a background. Without this, objects would blend into the background and appear artificial or poorly separated.

The Colour Palette also affects the final result, as all colours are first transformed before transparency is applied. For this reason, palette, shader, and background should always be considered together.

3.4.4 Start with Background

The option **Start with B00 background** is mainly useful when backgrounds are used without FX Manager.

Typical use cases:

- a single universal background for the entire game
- manual switching of backgrounds during gameplay

When enabled, background B00 is active immediately when the game starts. When disabled, backgrounds are initially turned off.

3.4.5 Background Images

Background images are stored inside the ZTG file and are not limited by ZX Spectrum colour restrictions.

As described in the user manual (Display → In-game backgrounds), images:

- can have significantly higher resolution than the original ZX image
- are automatically scaled to fit the game area

The recommended resolution is 512 × 384.

Higher resolutions can be used, but usually provide no visible benefit due to scaling. For best results, use resolutions proportional to the ZX Spectrum resolution (256 × 192).

3.4.6 Practical Usage and Design Approaches

The main goal of backgrounds is to replace large uniform areas with visually rich content.

There are two main design approaches.

Scene-Matched Backgrounds

Backgrounds are designed specifically for each scene, level, or menu.

In this approach:

- background elements are aligned with the game graphics
- additional details can be placed so that they appear as part of the original game

For example:

- extending walls, floors, or structures
- adding decorative elements that match the game style

This approach produces the most convincing results, as the background becomes an extension of the game itself.

Atmospheric Backgrounds

A more general background is used to enhance depth and visual quality.

In this approach:

- the background does not depend on exact object placement
- it adds texture, lighting, or atmosphere

Examples:

- gradients
- abstract textures
- environmental patterns

This is easier to create and works well across multiple scenes.

Combined Approach

Both methods can be combined.

For example:

- scene-specific elements for key areas
- general textures for the rest of the screen

3.4.7 Summary

In-game Backgrounds are one of the most visually impactful FX components.

They allow:

- replacement of flat ZX background areas
- integration of high-resolution artwork
- significant increase in visual richness

When combined with Colour Palette and Edge Colour Shader, they enable a wide range of visual styles while preserving the original gameplay.

3.5 FX Manager

FX Manager is an advanced system that allows FX parameters to change automatically during gameplay. Instead of applying a single visual configuration to the entire game, FX Manager dynamically switches background images, colour palette, edge shader parameters and foreground transparency based on what appears on the screen.

This allows different visual styles to be applied to menus, gameplay sections or individual levels, while the original game continues to run unchanged.

(Figure 8– FX Manager Edit window)

3.5.1 Basic Principle

FX Manager works by detecting predefined screen markers and triggering corresponding actions. A marker is a small fragment of the ZX Spectrum image captured at a specific screen position.

Detection is performed on every frame. When a marker is detected, the assigned action is executed and remains active until another marker is detected.

3.5.2 Markers and Actions

FX Manager supports up to 50 markers, each linked to one action. In addition, there is a special action called **At Start**, which is not linked to any marker and defines the initial background and FX slot when the game starts.

Each action defines which background will be used, which ZX colours will be treated as transparent, and which FX slot will be applied.

3.5.2a Practical Example: Menu and Gameplay Switching

To make the concept of markers, actions and slots concrete, here is a minimal example of setting up FX Manager for a typical arcade game that has a title screen followed by gameplay.

Goal: Show a stylised title background on the menu screen (Foreground Transparency 100%, background B00 fully visible), and switch to a gameplay background with black as the transparent colour and a dark shader style when gameplay starts.

Step 1 — Prepare the visual styles and store them into slots.

Configure the title screen look: set Foreground Transparency to 100% and enable backgrounds. Since the ZX Spectrum image will be completely invisible, the Colour Palette and Edge Colour Shader settings have no effect for this slot — only the background image will be visible. Open FX Manager, select slot 0, press Store to Slot. Slot 0 now holds the title screen visual style.

Now configure the gameplay look: set Foreground Transparency to 0%, set the transparent colour in In-game Backgrounds to black, adjust Edge Colour Shader with a Classic style, change Colour Palette to Classic ZX. Select slot 1, press Store to Slot. Slot 1 now holds the gameplay visual style.

Step 2 — Define the At Start action.

In FX Manager, select action At Start. Set Background to B00, set FX Slot to 0 (title screen style), set Colour to black (transparent). Press Apply. This defines what happens when the game first starts.

Step 3 — Capture a marker for gameplay start.

Start the game, play until the gameplay screen appears, then pause. Open FX Manager and enable the Game screen cursor option. Press the Settings button (top-left corner) to switch back to the emulator view — a blinking cursor will appear on screen. Use the left D-pad to move the cursor over a stable element that is unique to the gameplay screen, for example the score label or a fixed HUD element. Press the Settings button again to return to FX Manager, then press Capture. The captured marker preview appears in the interface. Now set this marker's action: Background B01 (gameplay background), FX Slot 1 (gameplay style), transparent colour black. Press Apply.

Step 4 — Test.

Start the game. The title screen should display with the full background visible. Once gameplay begins and the marker is detected, the system switches to slot 1 with the gameplay background. If the transition does not happen, verify that the captured marker area is stable and unique to that screen. If needed, switch to a different marker position or add Attribute Data detection for more reliable matching.

This two-slot, two-action setup is the foundation of all FX Manager configurations. More complex games simply add more slots and more markers following the same pattern.

3.5.3 FX Slots

FX Manager provides 10 slots (numbered 0 to 9) for storing complete FX configurations. Each slot holds a full set of parameters: Colour Palette, Edge Colour Shader, Foreground Transparency, and FX enable states.

Slots are passive containers. They do not change automatically. To store the current visual configuration into a slot, press **Store to Slot**. To load a slot back into the current parameters, press **Restore**. Both buttons are located at the bottom of the FX Manager screen. Slots are referenced by their index number in actions. When an action is triggered, the slot it references is immediately loaded into the active display parameters.

A typical approach is to configure each visual style you need (for example, one for menu screens, one for gameplay, one for each level), store each into a numbered slot, then assign those slot numbers in the corresponding actions. Section 3.5.13 describes working with slots in detail.

3.5.4 What an Action Contains

Each action consists of three parameters: background index, transparent colour or colours, and FX slot index.

The background index selects the active background image or disables it. Transparent colours define which ZX colours become invisible and reveal the background. The FX slot selects a complete configuration including Colour Palette, Edge Colour Shader, Foreground Transparency and FX enable states.

When a marker is detected, all these parameters are applied immediately.

3.5.5 Marker Detection

Markers are detected within a defined screen region of size 8×8, 16×8 or 8×16 pixels.

Each marker is defined by its screen position, size and data content. Detection can use pixel data, attribute data or both.

Pixel data represents the actual pixel pattern where 1 defines ink and 0 defines paper, while attribute data defines which colours are assigned to those areas. Detection is position-dependent, meaning that identical graphics appearing elsewhere on the screen will not trigger the marker.

3.5.6 Pixel and Attribute Detection

Two detection modes are available: Use Pixel Data and Use Attribute Data, and both can be enabled at the same time.

Pixel data provides precise shape matching, while attribute data provides colour-based matching. Using both modes together improves reliability, especially in cases where the same layout is reused but colours change.

3.5.7 Marker Priority and Search Behaviour

Markers are stored in a list and their position defines their priority, where lower index markers have higher priority.

The search does not always start from index 0. This behaviour is required to support both marker sequences and priority-based detection. By continuing the search from the last detected marker, the system can advance through sequences of identical markers. By returning to index 0 when needed, it preserves a clear priority system where lower index markers take precedence.

Detection is performed on every frame using a circular search. The system checks markers in order, starting from a defined position and continuing forward through the list, wrapping around to the beginning if needed.

When a marker is detected, its index becomes the reference point for the next frame. This allows the system to continue searching efficiently instead of always starting from the beginning.

The behaviour differs depending on whether the marker is used independently or as part of a sequence of identical markers.

If the next marker in the list is the same (a sequence of identical markers), the search in the next frame continues from that position. This allows progression through the sequence.

If the next marker is different, the detected marker is treated as a standalone marker and the next search starts again from index 0. This ensures that standard markers always follow a clear priority system.

3.5.8 Sequences of Identical Markers

Multiple actions can use the same marker to create a sequence.

This is useful in situations where the same visual element appears repeatedly, such as level transitions. When such a marker disappears and appears again, FX Manager advances to the next action in the sequence instead of repeating the previous one.

This allows multiple levels to share the same marker while still using different backgrounds or FX configurations.

If a marker outside the sequence is detected, the sequence is interrupted and the system returns to priority-based behaviour starting from index 0.

3.5.9 Practical Usage

Markers are typically selected from stable elements such as score displays, level numbers, HUD areas or static text.

A common approach is to define separate markers for intro screens, gameplay and game over screens. For level-based games, a repeating marker can be used to automatically advance through a sequence of actions, allowing each level to use a different background without needing a unique marker.

3.5.10 Capturing Markers

Markers must be captured directly from the game screen.

Start the game and pause the emulation at the desired moment. Open FX Manager and enable the **Game screen cursor** option. Press the **Settings button (top-left corner)** to switch back to the emulator screen. A blinking cursor will appear.

The cursor is moved using the **left D-pad**. The marker size is changed using the **right button on the right D-pad**, cycling through 8×8, 16×8 and 8×16.

After positioning the cursor over the desired area, press the **Settings button again** to return to FX Manager and press **Capture**. The captured marker preview appears in the interface.

Important: do not exit using the Back button, as this will remove the cursor. Navigation between FX Manager and the emulator screen should always be done using the Settings button.

3.5.11 Applying Changes

After modifying any action parameter such as background, transparent colours or slot index, the **Apply** button must be pressed.

If Apply is not used, changes will not be stored when switching to another action.

3.5.12 FX Slots — Full Reference

Section 3.5.3 provides a practical introduction to slots. This section covers the complete slot behaviour including important edge cases.

FX Manager provides 10 slots for storing complete FX configurations.

Each slot stores a full set of FX parameters:

- **Colour Palette**
- **Edge Colour Shader**
- **Foreground Transparency**
- **FX enable states**

Slots are used by actions to quickly switch between different visual styles during gameplay.

3.5.13 Working with Slots

Slots are passive storage containers and do not change unless explicitly updated by the user.

At the bottom of the FX Manager screen there is a Slot Index selector and control buttons.

- **Store to Slot** Stores the current FX parameters into the selected slot.
- **Restore** Loads the configuration from the selected slot into the current FX parameters.

This is an important concept: slots do not update automatically. The current FX parameters and slot contents are separate.

To modify a slot:

1. **Select the slot index**
2. **Press Restore (this loads the slot into current parameters)**
3. **Adjust Colour Palette, Shader or Transparency**
4. **Press Store to Slot to save changes back into the slot**

Slots can also be copied:

1. **Restore from one slot**
2. **Store to another slot**
3. **then adjust if needed**

3.5.14 Deleting Data

Several delete functions are available in FX Manager:

- **Delete All** Removes all markers and actions.
- **Trash icon (action area)** Deletes the currently selected marker and its action.
- **Trash icon (slot area)** Clears the selected FX slot.

3.5.15 Saving FX Manager Configuration

FX Manager configuration is not part of standard console parameters.

There are two ways the configuration can be stored:

- **Temporary storage – (Save Config / Load Config)**

  Used during editing. The configuration is stored outside the ZTG file and is preserved across restarts, allowing work to continue without updating the ZTG file.

- **ZTG file storage –** Used for persistent storage and final distribution.

All FX Manager data can be stored inside the ZTG file. When a game is started from a ZTG file, the console always loads FX configuration from the ZTG file and applies it to the current parameters.

This means that any unsaved changes made in FX Manager will be lost on the next start, because the ZTG file overwrites the current configuration.

For this reason, it is important to update the ZTG file whenever you want to test how the configuration behaves from a clean start or preserve the current state permanently.

3.5.16 Storage and ZTG Integration

All FX Manager data (markers, actions and slots) are stored in a dedicated FX block inside the ZTG file.

This block is created only when:

- **FX Manager is enabled**
- **and the ZTG file is updated with current console parameters**

It is important to understand that FX parameters exist in two forms:

- **Current FX parameters**

  These are standard console parameters (palette, shader, transparency, background colour). They can be used even without FX Manager.

- **Slot parameters**

  Stored inside FX Manager slots and used by actions.

When an action is triggered, the parameters from the selected slot are copied into the current FX parameters.

This means that slot parameters do not directly override current parameters continuously. Instead, they are applied at the moment the action is executed.

As a result, the current FX parameters always reflect the last activated slot.

For example, if the game is paused at any point, the current FX parameters will match the parameters of the slot that was last applied.

The FX Manager enable state itself is part of standard console parameters and is saved together with them.

When updating a ZTG file:

- **current console parameters are written into the standard parameter block**
- **if FX Manager is enabled, all FX Manager data are also written into the FX block**

3.5.17 Summary

FX Manager enables dynamic visual transformation of ZX Spectrum games by combining marker detection, action mapping and slot-based configurations.

It allows different visual styles to be applied automatically during gameplay, including scene-based backgrounds, level progression and visual transitions, without modifying the original game.

3.6 FX Quick Controls (Fn Shortcuts)

ZX Touch supports a Function key (Fn) that can be mapped to one of the eight physical buttons. The Fn key enables two different shortcut modes on the opposite D-pad, depending on whether it is pressed alone or together with the nearest adjacent button on the same D-pad. The Fn key assignment is configured in Settings — Physical Keys.

When only the Fn key is held (without the adjacent button), the opposite D-pad provides the standard console shortcuts:

Up / Down adjusts the volume.

Left / Right cycles through background images.

When Fn is held together with the nearest adjacent button on the same D-pad, the opposite D-pad provides two FX-specific shortcuts:

3.6.1 FX Master Toggle (Left)

Pressing Left on the opposite D-pad while holding the Fn combination toggles all FX features on and off simultaneously. This acts as a master switch that temporarily blocks or unblocks the Colour Palette, Edge Colour Shader, Foreground Transparency and In-game Backgrounds without changing any menu settings.

This is the most useful shortcut during FX development. It allows instant A/B comparison between the original ZX Spectrum image and the FX-enhanced version at any moment during gameplay, without opening a single menu.

3.6.2 Colour Palette Preset Cycling (Up / Down)

Pressing Up or Down on the opposite D-pad while holding the Fn combination cycles through the available Colour Palette presets. This allows rapid visual comparison of all palette styles directly during gameplay without opening the Display Settings menu.

This is particularly useful in the early stages of FX creation when deciding which colour style suits a specific game. Instead of switching between the settings menu and the emulator repeatedly, you can cycle through all presets while the game is running and settle on the one that works best.

3.6.3 Screen Capture Shortcut (Right on left D-pad)

When the Fn shortcut mode is active, pressing the **Right button on the left D-pad** triggers a screen capture. A confirmation message appears briefly in the top-left corner of the screen on a yellow background. The Screen Capture feature and its output formats are described in detail in section 3.7.

3.6.4 FX Status Indicator

When at least one FX option is enabled in the Display Settings menu, a small FX indicator appears in the top-right corner of the emulator screen. This indicator reflects the current FX state:

FX indicator visible — at least one FX feature is currently enabled and active.

FX indicator crossed out — FX has been temporarily disabled using the Fn master toggle shortcut. The menu settings are unchanged; FX will return to its previous state when toggled again.

FX indicator not shown — all FX options are currently disabled in the menus.

During FX development, the indicator provides a quick way to confirm whether FX is active at any given moment without opening the settings menu.

3.7 Screen Capture

Screen Capture is a separate function from the FX components, but it is one of the most useful tools during FX creation. It captures the current emulator frame and saves it to the SD card, where it can be accessed via FTP or by removing the card and connecting it to a PC.

3.7.1 Enabling Screen Capture

Screen Capture is enabled in the Display Settings menu. Three output formats are available and can be selected independently: **PNG**, **JPG** and **BMP**. Any combination of formats can be enabled at the same time. When multiple formats are selected, a single capture creates one file in each of the selected formats, all sharing the same sequence number.

3.7.2 What Each Format Contains

PNG is the most useful format for FX work. The content of the PNG depends on whether a background is currently active. If no background is active, the PNG contains a clean 512×384 image of the emulator output including all current palette and shader processing. If a background is active, the PNG contains only the foreground image with background areas fully transparent (alpha channel). This transparent PNG is ideal as a reference layer when designing backgrounds in an external image editor, as it shows exactly where game graphics will appear and allows precise alignment of background artwork.

JPG and BMP capture the composite image as it appears on screen, including the background if one is active. JPG uses lossy compression and produces smaller files, while BMP is uncompressed and produces larger files with no quality loss. These formats are useful for documenting results or testing visual appearance, but PNG is generally preferred for background design work because of the transparency support.

3.7.3 File Location and Naming

Captured files are saved in the **SD:/cap** directory. If this folder does not already exist, the console creates it automatically on first use.

Files are named automatically using a sequential numbering format, for example **cap_0003.png** and **cap_0003.jpg**. The number is always incremented from the highest number already present in the directory, regardless of which formats are currently enabled. This ensures that captures from different sessions never overwrite each other.

3.7.4 Practical Use in FX Creation

The typical workflow when using Screen Capture for background design is as follows. First, configure the palette and shader settings to the desired style. Enable a solid-colour temporary background that does not match any game colour (for example dark green if the game uses black). Run the game and take a PNG screenshot at a representative gameplay moment. Transfer the PNG to a PC via FTP (see section 4.1). Open the PNG in an image editor — the game graphics appear as a foreground layer with the solid background

colour easily removable by colour selection. Design the final background artwork using this layer as a placement guide. Once the background is ready, transfer it back to the console via FTP and update the ZTG file.

If a real background is already active when the capture is taken, the PNG contains only the foreground with full transparency, which is directly usable as a composition reference without any background removal step.

4. FX Workflow

This section describes a practical workflow for creating and testing FX configurations. It focuses on how to work efficiently with the system in real use, rather than explaining internal details.

It is assumed that the user is already familiar with basic console operation. If needed, general behaviour and controls are described in the User Manual.

4.1 Navigation During FX Preparation

Working with FX requires frequent switching between three parts of the system: the Game Browser, the Settings menu and the emulator itself.

The console is designed to make this switching fast and predictable. Two buttons are used for navigation: the Game Browser button in the top-right corner and the Settings button in the top-left corner. These buttons act as toggles and are mutually exclusive.

If the Game Browser button is active, the console displays the file browser. If the Settings button is active, the Settings menu is shown. If neither button is active, the emulator is running.

Each of these environments remembers its last state. When switching between them, the system always returns to the exact position where it was left. This allows the user to move between browsing files, adjusting FX parameters and testing gameplay without losing context.

In practice, this enables a smooth workflow. The user can open FX settings, adjust parameters, switch to the Game Browser to start the game, observe the result in the emulator, and then immediately return to Settings to continue refining the configuration.

4.1a File Transfer via FTP

During FX creation, frequent file transfers are needed between the console and a PC — copying background images to the SD card, retrieving PNG screenshots, and updating ZTG files. The ZX Touch built-in FTP server makes this significantly faster than removing and reinserting the SD card each time.

The FTP server is located in the Miscellaneous section of the Settings menu. For detailed instructions on connecting to a WiFi network and starting the FTP server, refer to the User Manual. Once connected, the console's IP address is shown on screen. Connect to this address using any FTP client on the PC (for example Total Commander, WinScp or FileZilla). The entire SD card is accessible as a standard file system, allowing files to be uploaded, downloaded, renamed and deleted without interrupting the emulator session.

A practical FTP workflow during background design looks like this: take a PNG screenshot using the Fn shortcut, connect the FTP client, download the file from SD:/cap, design the background in an image editor using the PNG as a reference layer, upload the finished background image to the game's directory on the SD card, then update the ZTG file and restart the game to see the result. The entire cycle can be completed without touching the console hardware.

4.2 Behaviour When Starting Different File Types

All FX parameters and FX Manager data are always active in memory, and every change is applied immediately.

When starting standard game files such as TAP, TZX, Z80 or SNA, the system does not clear FX data. Instead, it restores the FX enable states to the values stored in the user’s default configuration.

In most cases, this means that all FX features (palette, shader, transparency and FX Manager) become disabled, because they are typically turned off in the default configuration.

Importantly, this process does not modify any FX parameters themselves. Palette settings, shader configuration and all FX Manager data (markers, actions and slots) remain unchanged in memory.

After this initial step, the started game may apply its own stored configuration, if such data exists. ZX Touch allows storing per-game parameters, and these can override FX parameters such as palette, shader and enable states.

However, FX Manager data is not part of this system and cannot be modified by standard game files. It exists separately and can only be affected by Load Config or by a ZTG file that explicitly contains FX Manager data.

In summary, starting a standard game file only affects which FX features are currently enabled, but does not remove or overwrite the underlying FX Manager configuration.

4.3 ZTG File Structure and FX Data

ZTG files can contain two independent types of data: standard console parameters and FX Manager data (markers, actions and slots). These are stored as separate blocks inside the file.

A ZTG file may contain only standard parameters, both standard parameters and FX Manager data, or no parameters at all.

When a ZTG file is created or updated, the stored content depends on the current state of the system. If parameter saving is enabled, the standard parameter block is written into the file. If FX Manager is enabled at that moment, an additional FX Manager block is also created and stored.

This means that FX Manager data is included in the ZTG file only when it is explicitly active during the save process.

This separation ensures full backward compatibility. ZTG files created before the FX system continue to work exactly as before, because they only contain standard parameters and do not include any FX Manager data.

4.4 RAM-Based Workflow (Working Without ZTG Parameters)

During development, it is often useful to work without any parameters stored inside the ZTG file.

To achieve this, the ZTG file must be prepared without any parameter blocks. In this case, the file does not contain any stored console configuration.

All other ZTG content can still be present, including the game itself, Title image, Info image, Info text, background images, poke file and selected emulation mode (48K or 128K).

When such a ZTG file is started, it does not modify the current state of the console in any way. No parameters are loaded, and nothing is overwritten.

This means that all current settings remain exactly as they were before starting the game, including FX parameters, FX Manager configuration, key mappings and all other console settings.

Unlike standard file types, this type of ZTG file also does not disable FX Manager. The system continues to operate exactly as it was configured.

This behaviour enables a RAM-based workflow, where all work is done directly in memory. The user can adjust FX parameters, define markers and actions, restart the game and immediately see the results, without saving anything into the ZTG file.

The configuration can be preserved using Save Config and restored later using Load Config.

Once the work is complete, the final configuration can be written into the ZTG file by enabling parameter saving and updating the file.

4.5 ZTG-Based Workflow (Working With Stored Configuration)

An alternative approach is to work directly with ZTG files that contain full FX configuration.

In this workflow, all FX parameters and FX Manager data are stored inside the ZTG file. Each time the file is started, the entire configuration is loaded into memory, including markers, actions and slots.

This ensures that the game always starts in a fully defined state, exactly as intended.

When using this approach, it is important to understand that starting a ZTG file always overwrites the current state in memory. Any unsaved changes will be lost.

Because of this, the typical workflow is based on repeated updates of the ZTG file.

After making changes to FX parameters, markers or slots, the user updates the ZTG file to store the current configuration. The game is then restarted to test the behaviour from the beginning.

This approach is slower than the RAM-based workflow, but it provides a reliable way to verify how the final result behaves when the game is launched.

It is especially useful in later stages of development, when the configuration is already stable and needs to be tested as a complete package.

4.6 Choosing Between Workflows

Both workflows are valid and can be used depending on the situation.

The RAM-based workflow is faster and more flexible. It is ideal during development, when parameters are frequently adjusted and markers are being created.

The ZTG-based workflow is more controlled. It is useful when verifying the final behaviour of the game and ensuring that everything works correctly from startup.

In practice, most users will combine both approaches. Initial work is typically done in RAM, using Save Config and Load Config, while final testing and distribution are done using ZTG files.

4.7 Quick Decision Reference

The following situations are common during FX development. Use this as a quick reference when you are unsure which approach to take.

I want to enhance a single game with one consistent visual style throughout. Use Colour Palette, Edge Colour Shader, Foreground Transparency and one Background. Enable Start with B00 background. You do not need FX Manager at all. Save the configuration into the ZTG file.

I want different backgrounds for different game screens or levels. Use FX Manager with markers and actions. Store each visual style into a separate slot. Define one marker per screen transition. For level-based games, consider using identical markers in a sequence so that one marker advances through all levels automatically.

The marker I captured is not being detected reliably. Try enabling both Use Pixel Data and Use Attribute Data. Move the marker to a different screen position — prefer areas with fixed text, score displays or static HUD elements that do not change during gameplay. Avoid areas near sprites or moving objects.

My objects look pasted over the background with no natural separation. The Edge Colour Shader outline is likely missing or too subtle. For the transparent ZX colour, set both L1 and L2 to the same colour as your intended outline. Make sure Standard shader mode is active. Check that the shader is enabled at all.

I want the loading screen or title screen fully replaced by my own artwork. Set Foreground Transparency to 100% for the action that covers that screen. The ZX Spectrum output will be invisible and only your background image will show. The game continues to run normally.

My changes are lost every time I restart the ZTG file. This means your ZTG file contains stored parameters that overwrite the current configuration on start. Switch to the RAM-based workflow: update the ZTG file without parameter blocks so it stops overwriting. Use Save Config to preserve work between sessions.

I want to test a slot at a specific level without replaying from the beginning. Load a Z80 or SNA snapshot that puts you directly at that level. This disables FX Manager but all slot and marker data remains in RAM. Capture markers and adjust slots in this state, then return to your ZTG file. FX Manager can be re-enabled without losing your configuration.

5. Getting Started with FX Creation

This section explains how to begin creating FX configurations from scratch. The goal is to guide you through a practical workflow that leads to a working result, not just to describe individual features.

5.1 Preparing the First ZTG File

Before working with FX, create a basic ZTG file that will serve as your working project. The file should contain the game itself and at least one background image. The background does not need to be final; its purpose is to allow immediate testing of transparency and shader behaviour.

At this stage, also configure basic console settings such as key mapping and any other preferences required for comfortable gameplay.

Depending on the workflow you plan to use, you can either save these settings into the console (Default or one of the user memory slots) or store them directly in the ZTG file. For the initial setup, it is recommended to work with ZTG-based saving.

FX Manager should remain disabled in this phase, and no FX Manager data should be stored.

5.2 Initial Visual Setup (Without FX Manager)

In the first phase, work only with the core FX components: Colour Palette, Edge Colour Shader, Foreground Transparency and Backgrounds. FX Manager should remain disabled.

Adjust parameters, update the ZTG file, restart the game and evaluate the result. Repeat this process until you achieve a stable visual style.

The goal is to establish one consistent look that provides good readability, clear separation between foreground and background, and a balanced use of colour. Do not try to solve every scene at once. This initial setup will later become the foundation for all FX Manager configurations.

5.3 Planning Before You Continue

Before enabling FX Manager, it is useful to briefly think about how the game should be structured.

Consider which parts of the game should look different, whether menus, gameplay and special screens need separate setups, and whether level progression can be handled as a sequence or requires individual markers.

Also consider whether some screens should be fully replaced using high transparency.

This does not need to be a detailed plan. A simple idea of the structure helps determine how many slots and markers will be needed and avoids unnecessary rework later.

5.4 Using PNG Screenshots for Background Preparation

PNG screenshots are an essential tool during FX creation.

If no background is active, the PNG contains a clean 512×384 image of the emulator output. If a background is active, the PNG contains only the foreground image, while background areas are fully transparent.

The captured image already includes palette adjustments and shader processing, making it ideal for external editing.

A practical workflow is to prepare a rough palette and shader setup, enable a temporary background, take a PNG screenshot and open it in a PC editor. The transparent image can then be used as a reference to design or refine the background.

This approach makes it much easier to align visual elements, match colours and achieve a clean integration between foreground and background. For full details on capture formats, file naming and the Fn shortcut used to trigger captures, see section 3.7. For transferring captured files to a PC without removing the SD card, use the FTP server as described in section 4.1a.

5.5 Introducing FX Manager

Once a stable visual setup is created, the next step is to automate it using FX Manager.

So far, all changes were static. FX Manager allows different configurations to be applied automatically during gameplay.

Instead of a single visual setup, the game can now have multiple configurations, for example one for menu screens, another for gameplay, and additional ones for levels or special scenes.

FX Manager works by connecting markers, actions and slots. In the following steps, this structure will be created.

5.6 Building the First FX Logic (Slots and Actions)

Start by storing your current visual setup into a slot. Select a slot and press Store to Slot. This saves the current palette, shader and transparency settings.

The first action that should be defined is At Start. This action determines which slot and background are active when the game begins.

After that, create a minimal structure with a small number of markers. A typical starting point is one marker for menu or intro screens and one marker for gameplay. Each marker should trigger a specific slot and background.

At this stage, keep the setup simple. The goal is to confirm that switching between configurations works correctly before expanding the system.

5.7 Working with Markers During Development

Markers can be captured at any time, independently of the current FX Manager state.

Disabling FX Manager does not erase markers, actions or slots. All data remains in memory.

This allows flexible workflows. For example, you can load a Z80 or SNA snapshot to jump directly to a later level, or use poke files or saved positions to reach specific game states. Although loading such files disables FX Manager, all configuration data remains intact.

You can capture markers in this state, then return to your ZTG file and continue working. This avoids repetitive gameplay and significantly speeds up development.

5.8 Switching to RAM-Based Workflow

Once the basic FX logic is working, it is often more efficient to switch to a RAM-based workflow.

This is done by preparing a ZTG file without parameter blocks. In this mode, starting the ZTG file does not overwrite current settings, and FX Manager remains active.

All parameters remain in RAM, allowing immediate testing and rapid iteration without rewriting the ZTG file.

Configuration can be stored and restored using Save Config and Load Config.

5.9 Finalising the Configuration

When the FX setup is complete, enable parameter saving and update the ZTG file.

This writes both standard FX parameters and FX Manager data into the ZTG file. After this step, the game becomes a fully self-contained FX package.

Restart the ZTG file to verify correct behaviour from a clean start, including marker detection and slot switching.

5.10 Practical Tips and Good Practices

A collection of fully prepared ZTG files is available on the official ZX Touch website. These can be used as reference examples. By examining existing configurations, you can see how slots, markers and actions are structured and use them as a starting point for your own work.

One of the most important aspects of FX design is achieving a balance between foreground and background. The background should support the game image, not compete with it. Highly detailed or photorealistic backgrounds can create a visual mismatch with ZX Spectrum graphics. In many cases, a simpler or stylised background produces better results.

Always consider readability, contrast and stylistic consistency.

When preparing a game for FX, it is recommended to include a Title image and an Info image. Adding a small “ZXT FX” mark helps indicate that the game has been enhanced using the FX system.

If you create a well-prepared configuration, consider sharing it. ZTG files can be distributed to other users, and selected examples may be included in the official collection.

5.11 Summary Workflow

A typical workflow follows these steps: create a base visual style, refine backgrounds using PNG screenshots, store configurations into slots, connect markers to actions, test and iterate in RAM, and finally save everything into the ZTG file.