PCA 3D ADVENTURE STUDIO

Styling Your GUI

Colors, corners, images, shaders and effects that work together

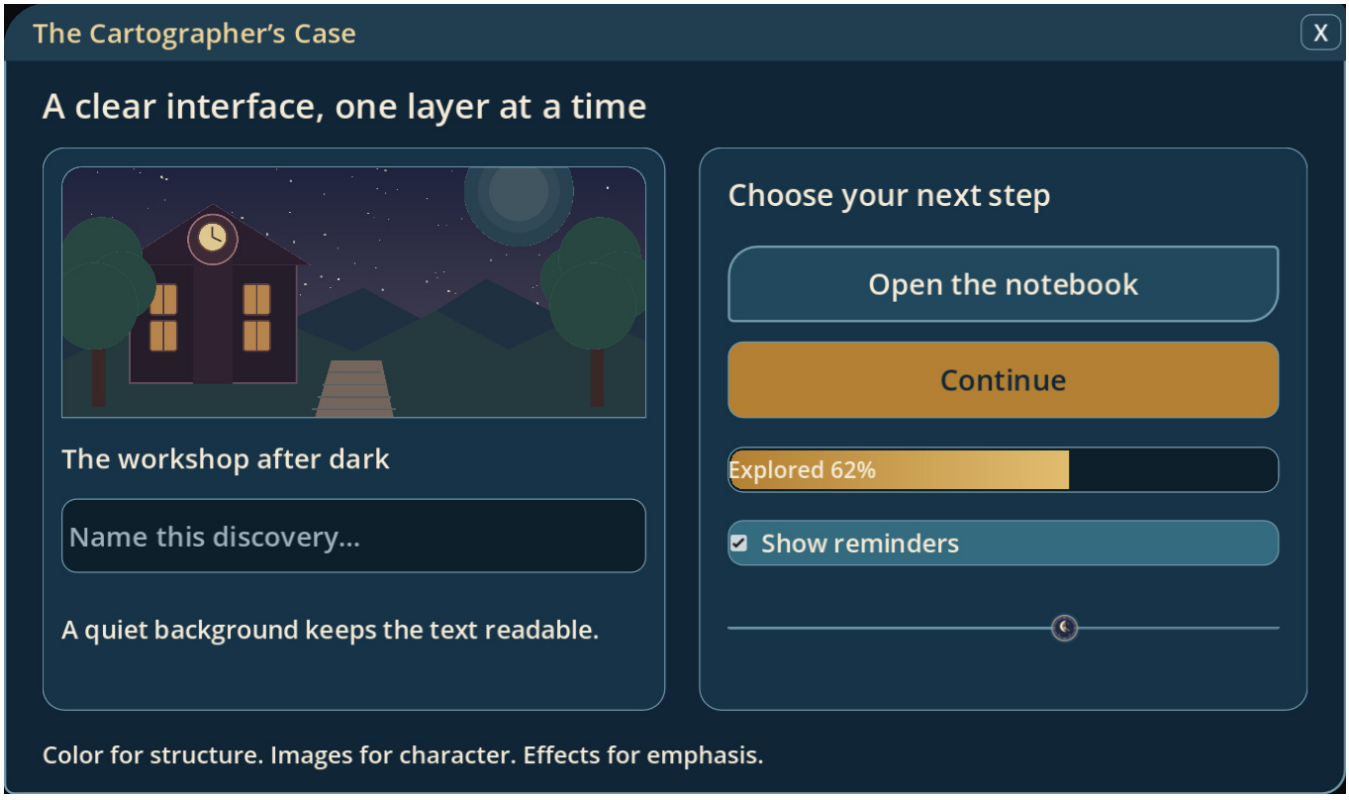

Figure 1. A complete interface using a small palette, separate rounded surfaces and restrained emphasis.

An in-depth practical guide for Windows, macOS and Linux.

Editor reference: 0.86.3.4 | Companion to the GUI Designer tutorials

Before you begin

This guide shows how to give an adventure game’s interface a consistent appearance in PCA 3D Adventure Studio. We will start with ordinary colors and clear text, then add independent corners, state images, decorative finishes, shader layers and opening effects.

The running example is The Cartographer’s Case: a small window with an illustration, a text box, two buttons, a progress bar, a checkbox and a slider. You will also see inventory and save widgets, a model viewer, and complete parchment and terminal variations.

All screenshots come from the editor or its running game preview. The shader comparisons use the same engine image so the differences are visible. They are examples of the available effects, not a suggestion to turn them all on at once.

You can follow along in your own project. No practice project or additional practice files are included. The instructions use the same desktop workflow across Windows, macOS and Linux.

You should know	This book adds
How to add a GUI and an element	How to style each surface without changing the wrong one.
The difference between a name and an ID	Repeatable colors, corners, images and shader settings.
How to assign a Button event	Examples using shared actions, visual nodes and scripts.
How to run a game preview	A deliberate way to compare appearances and test input states.

The reference build is 0.86.3.4. Normal GUI edits preview live; dialogs have their own Apply buttons, Raw JSON needs Validate and Apply, and script records need their own Apply Changes. Save Project after authoring.

Contents

Click a chapter title, or use the PDF bookmarks to open any continued page.

1. Choose the surface before the color 4
2. Build a clean starting window 7
3. Give each color a job 9
4. Understand transparency before using it 12
5. Make typography part of the style 15
6. Give corners their own shapes 17
7. Use images as decoration and content 21
8. Style feedback without changing behavior 24
9. Finish the special controls 27
10. Add a decorative surface finish 29
11. Apply your first element shader 31
12. Understand the shader controls 34
13. Pick an effect for a reason 37
14. Build a stack one layer at a time 45
15. Give the window an entrance and exit 48
16. Change styling during play 51
17. Build three complete looks 58
18. Troubleshoot color and shape problems 62
19. Exact color and corner keys 66
20. A final practical check 69

1. Choose the surface before the color

You selected…	You are styling…	Typical controls
GUI Window	The containing window and its chrome.	Window background, outer border, title bar, close control.
An ordinary element	That Button, Label, Panel, Image or other control.	Text, background, corners, image states and element shaders.
A composite widget	Its outer surface and its repeated internal parts.	Grid, slots, save screenshots, navigation and text styles.
The editor’s Theme button	The application workspace around your game.	Editor chrome, inspector controls and the authoring environment.

A GUI is built from several surfaces. The window is the outer container. A Panel can make a smaller card inside it. A Button can sit on that card. Each has its own color and corner settings.

Start by clicking the intended item in the GUI Elements list. Check the inspector heading before editing. To work on the window itself, use GUI Window in the designer toolbar. The inspector will then show the window’s properties instead of the last selected element.

> Putting a Button over a Panel does not make the Button share all of the Panel’s styles. Likewise, a shader on a Panel is not automatically a whole-window filter over every element you can see nearby.

Editor themes and the player’s interface

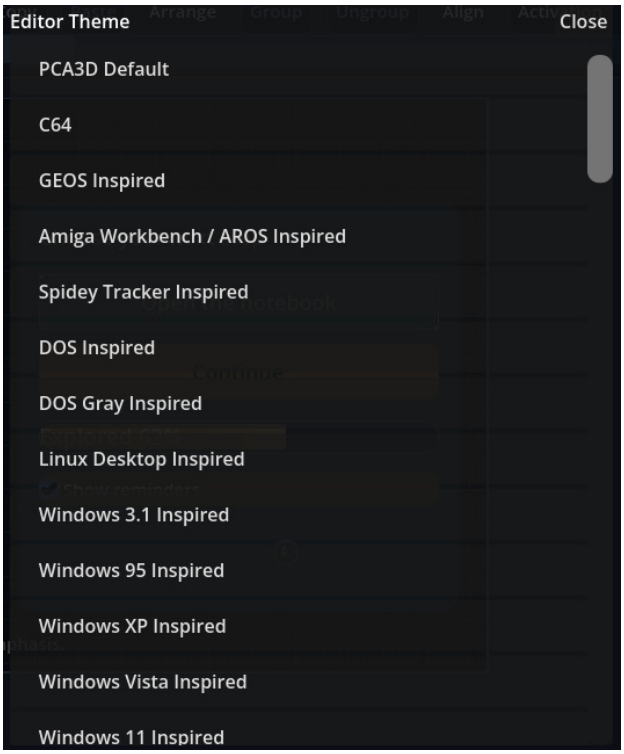

Figure 2. The Theme chooser styles the editor workspace. Your authored game interface has its own settings.

The top toolbar’s Theme control is useful for making the editor comfortable to work in. It is separate from the colors, images and shader stacks you author for a game GUI.

For a game’s visual theme, choose a repeatable palette and apply it to the GUI Window and its elements. Clone a finished menu when you need another window in the same style. Do not expect changing the editor theme to replace every authored button color in your exported game.

Some native controls can still use inherited theme resources for details you have not explicitly styled. Test the exported game’s actual controls, especially focus indicators and choice widgets, rather than judging only the surrounding editor chrome.

> Use an editor theme that lets you judge your game fairly. A bright workspace can make a dark menu look different from how it feels in a full-screen game.

Find the settings without hunting

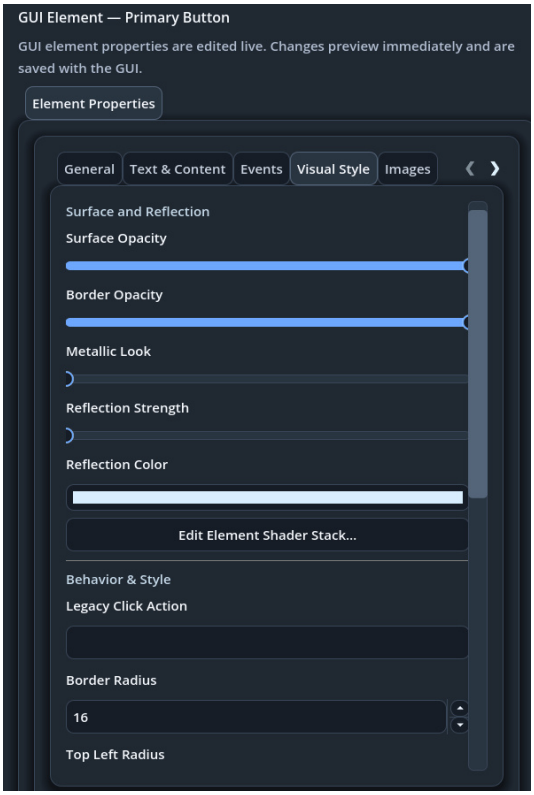

Figure 3. Check the selected element’s name, then choose the tab that owns the change.

Tab or toolbar control	Use it for
General	Position, size, anchor, Z Index, visibility and enabled state.
Text & Content	Caption, font, size, decorations, text color, text opacity and alignment.
Type Settings	Controls particular to this type: progress fill, media layout, slider behavior and widget parts.
Visual Style	Surface opacity, finish, corners and background/border colors.
Images	Normal, hover and clicked artwork, fit, and supported internal image regions.
Element Shader	The selected element’s layered effects.
Transitions	The GUI’s entrance and disappearance.

The tabs can extend beyond the visible inspector width. Use the small tab arrows. The scrolling inside a tab is separate from those tab arrows.

2. Build a clean starting window

1. Add a GUI named The Cartographer’s Case with ID style_lab. Keep a copy of an existing GUI before experimenting.
2. Select GUI Window. Use a window size of 1120 × 660 for this example. Place its content inside a comfortable 32-pixel margin.
3. Enable its background color and border. Start with background #102637, border #7199AA, border width 2 and border radius 20.
4. Add two Panels: X 32, Y 120, Width 520, Height 470; and X 580, Y 120, Width 508, Height 470.
5. Give the panels background #173347 and radius 18. Keep their text and controls inside those edges.

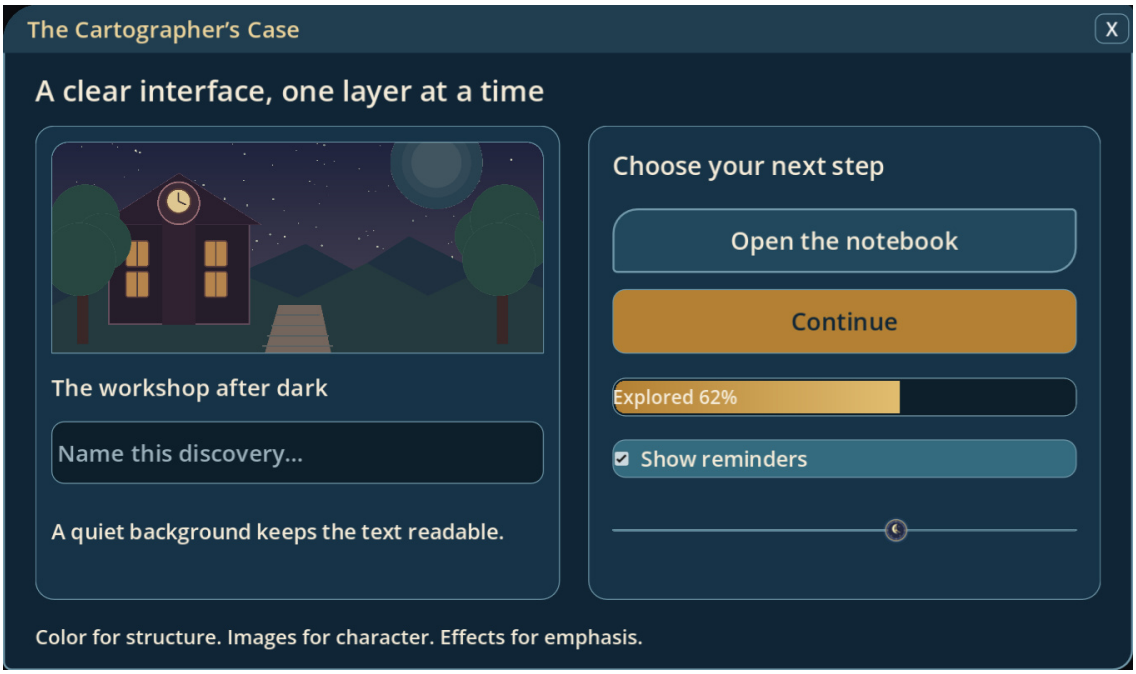

Figure 4. The starting layout separates illustration, input and primary choices.

> The dimensions are design coordinates, not a required monitor resolution. The example leaves space for a title bar and footer. Check the game’s actual presentation size before choosing your final font and button sizes.

Layering and spacing make the style work

Part	Practical starting rule
Background Panels	Put them behind content with a lower Z Index.
Images	Keep them above their backdrop and below any intentional text overlay.
Labels and Buttons	Give them an explicit higher layer when they overlap a Panel.
Margins	Leave room between a curved border and the text.
Gaps	Repeat a few gap sizes rather than inventing a new one for every element.

Z Index is in General. For a simple layered card, try Panel 0, Image 1 and text/controls 2 or 3. When overlapping elements share the same layer, do not depend on a lucky drawing order.

A border radius changes the shape you see. It does not add padding for the caption. If text crowds a round corner, move or resize the content; increasing the radius again usually makes the crowding worse.

Use alignment tools, snapping and repeated dimensions to keep related controls tidy. A 64-pixel Button next to another 64-pixel Button reads as one set. A collection of slightly different heights looks accidental even when the colors match.

> If text disappears behind an opaque Panel, inspect Z Index before changing text opacity or font color. Transparent panels can hide the same layering mistake by letting part of the text show through.

3. Give each color a job

Role	Midnight palette	Use
Window surface	#102637	The overall backdrop.
Panel surface	#173347	Cards that hold related controls.
Ordinary Button	#21485D	Secondary choices.
Hover	#356B7F	Pointer feedback.
Pressed	#825F3A	Temporary held-input feedback.
Border	#7199AA	Shape and separation.
Main text	#F5ECD9	Most captions and instructions.
Accent	#B48134	A primary choice or important fill.

A small palette gives you enough variety without making each element look as though it belongs to a different game. Reserve your accent for something that deserves attention.

Use the same ordinary, hover and pressed colors for similar Buttons. A destructive action can have a deliberate exception, but it should also have clear wording. Color alone is a poor explanation of what Delete does.

> Write down the palette values as you work. The guide’s values are a starting recipe; change them to suit your game’s artwork and lighting.

Use the color picker precisely

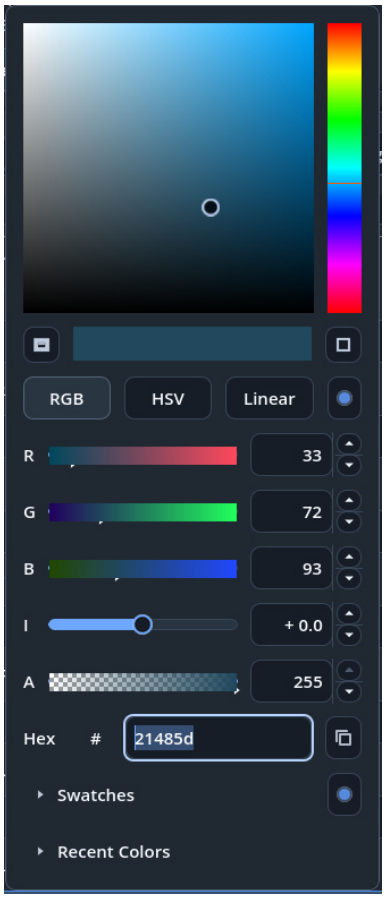

Figure 5. The color picker provides RGB values, alpha, a hex field, swatches and recent colors.

1. Click the color swatch next to the field you want, such as Background Color.
2. Choose visually with the large color area and hue strip, or enter an exact value in Hex.
3. For #21485D, enter 21485d in the hex field. With an opaque color, the alpha control should be at its full value.
4. Click outside the picker to return to the inspector. Inspect the actual element, not just the swatch.
5. Save a useful color as a swatch or reuse its written value for the next element.

RGB describes the color channels. HSV is an alternative way to choose hue, saturation and brightness. Alpha describes transparency. A vivid color with alpha zero is still invisible.

> The picker’s alpha display can use a channel value such as 255 for full opacity. The element’s Surface Opacity slider uses 0–1. Those are different numerical presentations of transparency.

Set normal, hover and pressed colors



Figure 6. Visual Style provides separate backgrounds and an outline color.

1. Select a Button, then open Visual Style.
2. Set Background Color to #21485D.
3. Set Hover Background to #356B7F.
4. Set Pressed Background to #825F3A.
5. Set Border Color to #7199AA. In Text & Content, use #F5ECD9 for the caption.
6. Use GUI Preview or Play to compare the normal, hovered and held states.

A Label’s text color belongs in Text & Content. Changing its background color is not the same edit. For a text-only Label, an alpha-zero background and zero border width keep the focus on the words.

> An opaque image can cover the background color. If a change seems ineffective, inspect Images for a background image or a Media Asset fallback before choosing a brighter color.

4. Understand transparency before using it

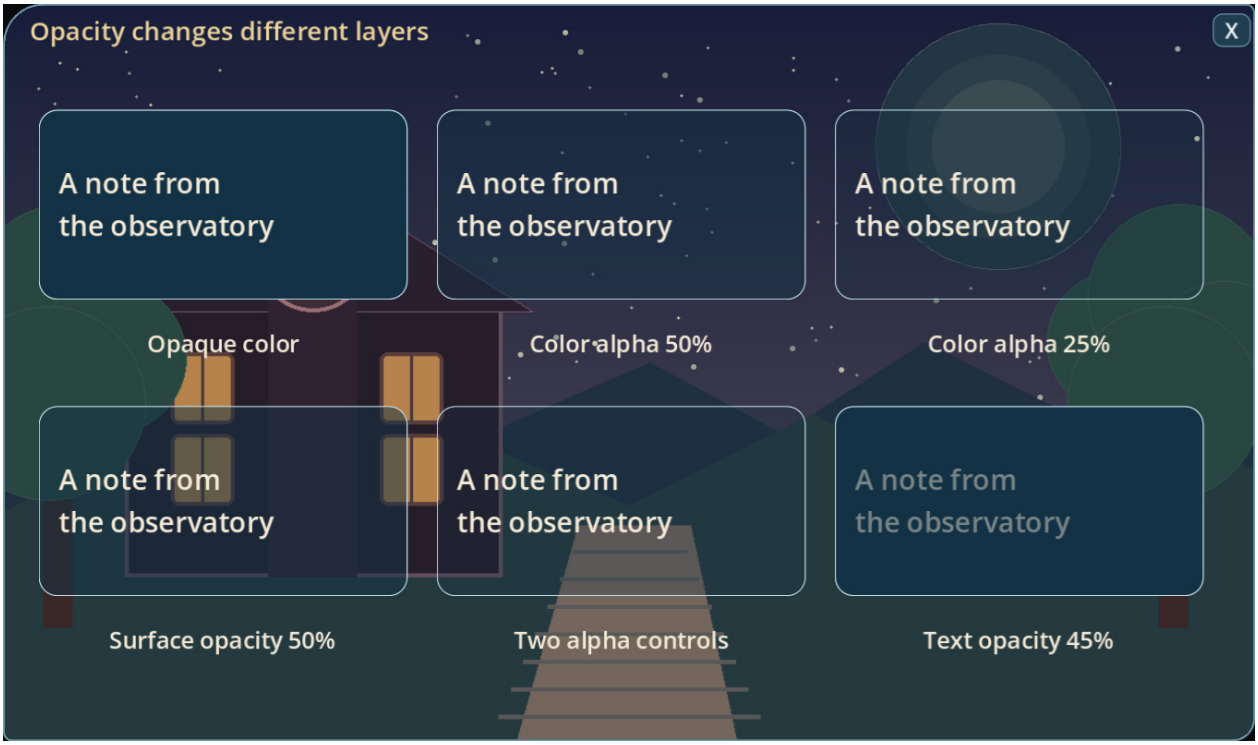

Figure 7. The same illustration shows through panels at different alpha levels. Text is kept on a higher drawing layer.

Hex form	Meaning
#RRGGBB	A color with full alpha.
#RRGGBBAA	Color plus alpha in the final two digits.
#143245FF	The panel color, opaque.
#14324580	About half-opacity color.
#14324540	About quarter-opacity color.
#14324500	The same color, fully transparent.

In this format the final pair is alpha. Do not put alpha at the front as though the value were AARRGGBB. That would change the color channels instead.

Surface, border and text opacity differ

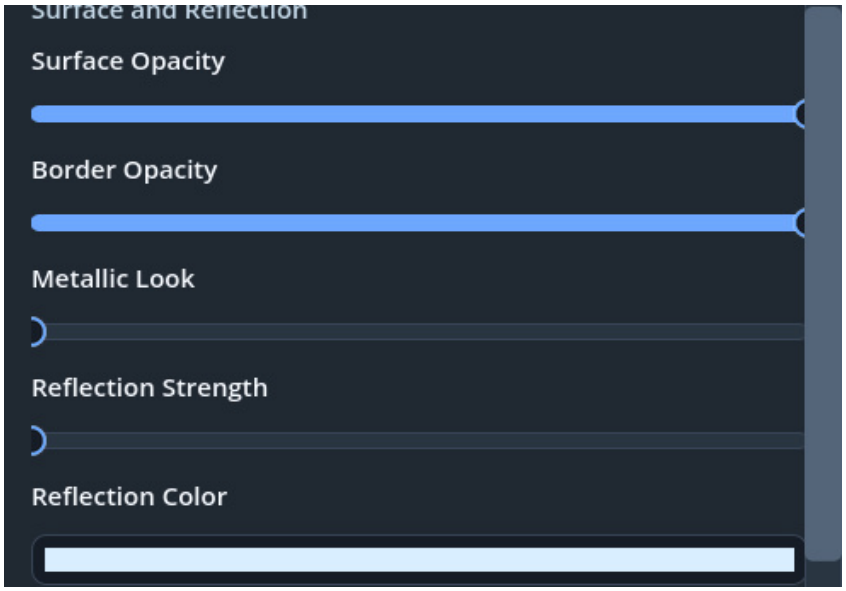

Figure 8. Surface Opacity and Border Opacity are separate. Text Opacity is in Text & Content.

Control	Use it to change
Surface Opacity	The ordinary surface/backdrop presentation supported by that element.
Border Opacity	The frame’s alpha, independently of the background.
Text Opacity	The caption’s alpha.
A color’s alpha	Transparency embedded in that particular color value.
A GUI transition	The temporary presentation of the window as it opens or closes.

Alpha controls can multiply. A background color at 50% alpha, drawn at Surface Opacity 0.5, contributes roughly 25% alpha. If a panel is unexpectedly faint, check both controls.

The demonstration uses separate Labels above the panel surfaces, which makes the difference easy to judge. A whole GUI fade or disabled-state modulation can affect a larger tree of controls; it is not the same operation as lowering one panel’s background alpha.

> A transparent background does not imply that the element is disabled or that all clicks pass through it. Input behavior is controlled separately.

Choose transparency that survives gameplay

A translucent menu can look excellent over one room and become unreadable over the next. Test it over a bright wall, a dark corner, a busy texture and a moving scene.

Problem	Useful adjustment
Text competes with the room	Raise the background alpha or add a quieter Panel behind that text.
The whole interface feels muddy	Reduce overlapping translucent surfaces. Use one clear window surface and lighter internal structure.
Borders disappear	Raise Border Opacity or choose an outline color with more separation.
Only text looks faint	Check Text Opacity and the text color’s alpha.
An invisible element still seems to block input	Inspect its enabled state and event/input behavior; transparency is a visual setting.

For a dependable first menu, use an opaque window. Introduce transparency after the text, spacing and interaction states work. A small translucent information card is easier to control than an entire nearly transparent save browser.

> “Glass” artwork and a low-alpha panel are appearance choices. They do not automatically produce a live blur of the room behind the GUI.

5. Make typography part of the style

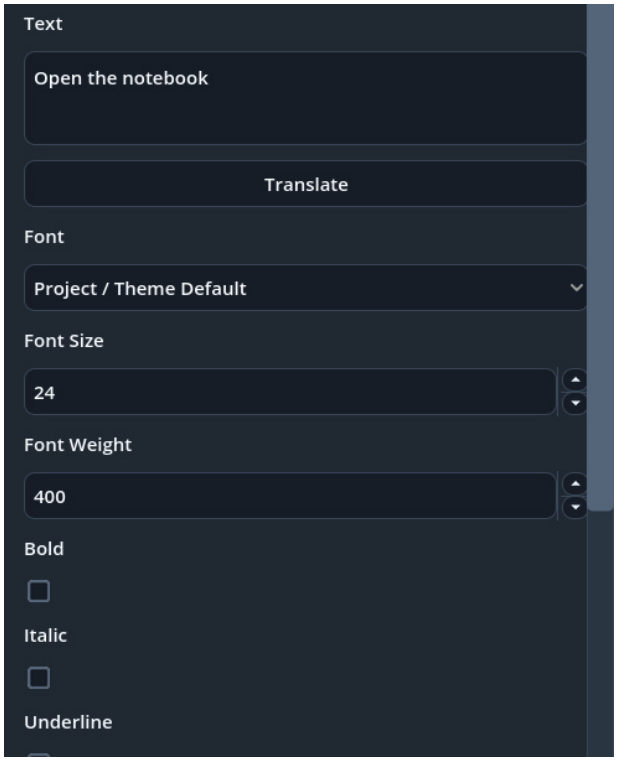

Figure 9. Text & Content owns the font and its decorations.

Choose a readable body font first. Use a modest size hierarchy: a larger window/section title, a normal Button caption and a smaller supporting note. The example uses roughly 29, 24 and 20–21 design pixels for those roles.

The font picker can use your available game font records and supported font choices. A stylized title font can work well, but small instructions need clear letter shapes. Test punctuation, numbers and every language your game supports.

Weight and Bold change emphasis. Italic, Underline and Strikethrough are additional decorations. Use them deliberately; underline often suggests an interactive link, while strikethrough usually means something is no longer available or relevant.

> A font’s available faces and weight support affect the result. Do not expect every font to have a convincing heavy or italic variation. Imported fonts also need the glyphs your translations use.

Alignment, placeholder text and long captions

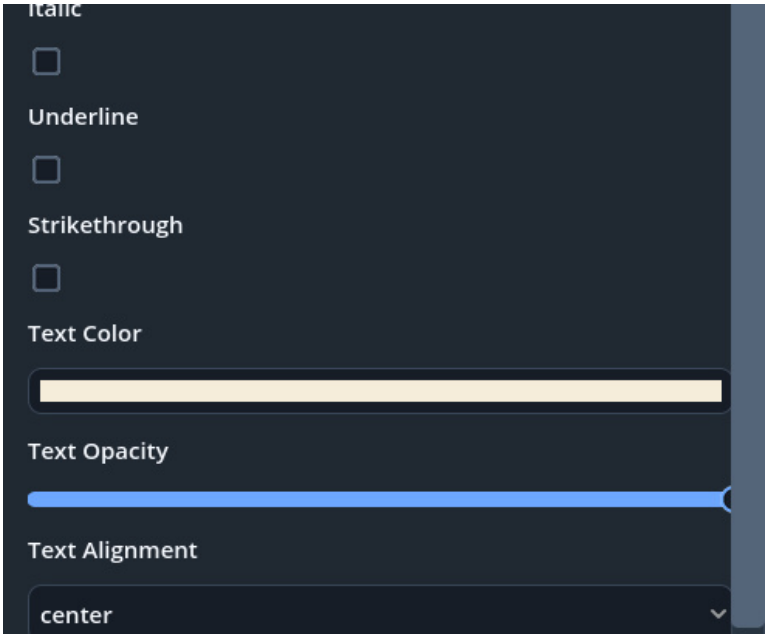

Figure 10. Text Color, Text Opacity and alignment are separate from the element’s surface.

Center short Button captions. Use left alignment for instructions, descriptions and most lists. Left-center and right-center are useful when text needs vertical centering inside a tall control.

A Text Box has its own placeholder appearance in Type Settings. The placeholder is a prompt before the player types; it should be readable without looking identical to entered text. The example uses #9CB2BC against a dark input surface.

Long translations need room. Test the longest actual caption in the intended font. Do not solve every overflow by making the text smaller: a wider Button, taller description or shorter caption can be the better edit.

> A shader on a Button can process its caption too. If a decorative treatment makes text hard to read, put the art on a separate element and keep the caption unshaded.

6. Give corners their own shapes

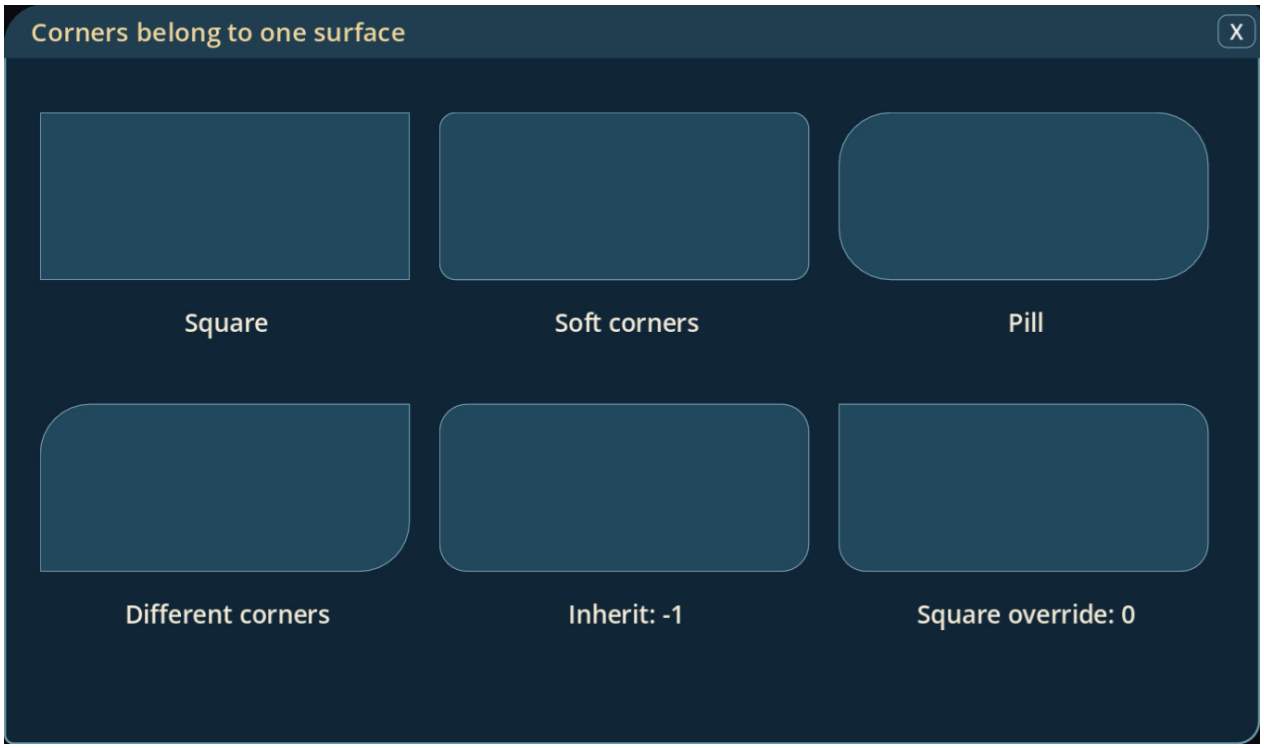

Figure 11. Square, soft, pill and asymmetric corners use the same basic controls.

Value	Meaning
Border Radius 0	Square overall corners.
A positive common radius	A curve applied where a corner has no explicit override.
Corner -1	Use this surface’s common radius.
Corner 0	Make that particular corner square.
Positive corner override	Use that radius for this corner only.

Top Left, Top Right, Bottom Right and Bottom Left refer to the visible rectangle. The workshop Button uses common Radius 16, Top Left 24, Top Right 4, Bottom Right 24 and Bottom Left 4.

> Large radii are constrained by the space available. A radius larger than half a short control’s height will not keep growing into an unlimited curve. Judge the visible shape at the intended size.

Edit the window and its title separately

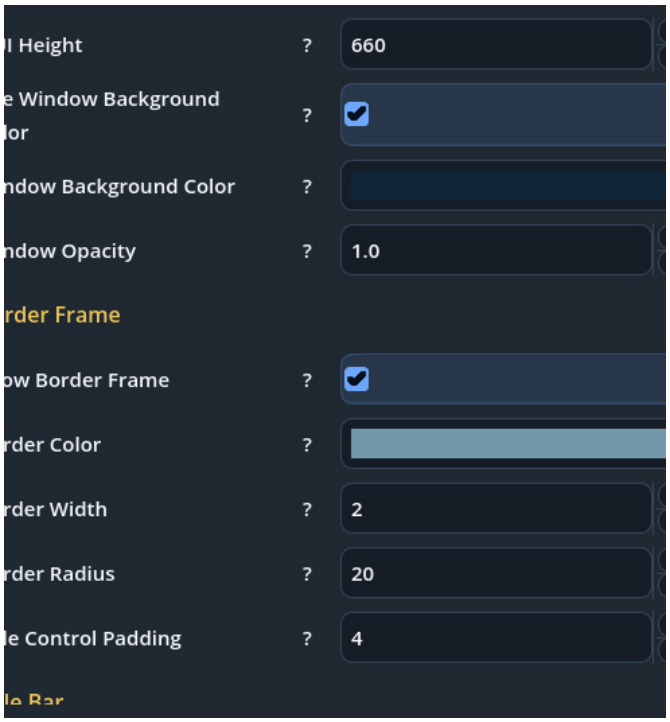

Figure 12. GUI Window has dedicated background, border and title-control settings.

1. Click GUI Window in the toolbar.
2. In Type Settings, enable Use Window Background Color and choose Window Background.
3. Enable Show Border Frame. Set Window Border Color, Width and Radius.
4. In Visual Style, set the four individual corner overrides.
5. Return to Type Settings for title-bar and close-button styling.

The window’s common radius is stored as windowBorderRadius. An ordinary element uses radius. The four corner keys use the same radiusTopLeft-style names within their respective records.

> Window title chrome is easiest to judge in Play. The designer’s grid and selection display can look different from the completed window background and close-control placement.

Title bars, close controls and consistent framing

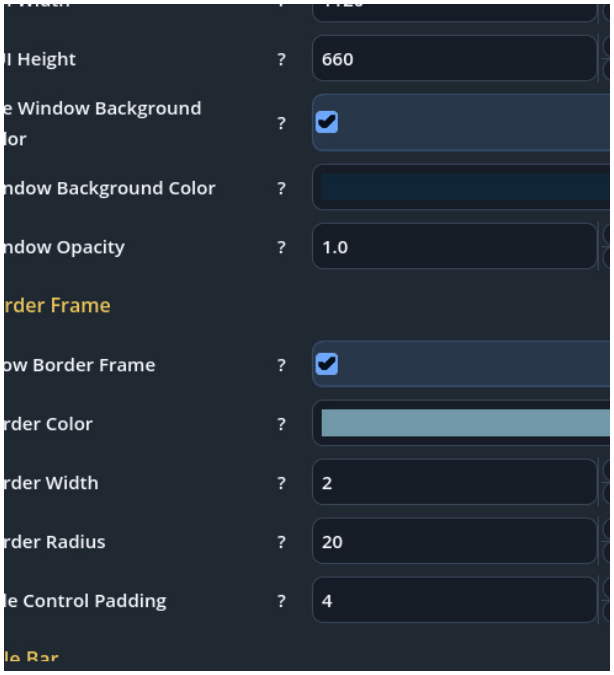

Figure 13. The title bar has its own background, text and font settings.

Use Show Title Bar when the window benefits from a clear heading or draggable chrome. Choose its height, background, caption font/color and alignment. Leave enough horizontal room for the close control and any title icon.

The Close Button has separate normal, hover and pressed backgrounds, text or image content, dimensions, border and its own common radius. Its shape is independent of the window’s outer corners.

Title icons and close controls have placement choices. In the example the close control is upper-right. Use a visible, generous hit area when the game supports touch, and confirm that an ornamental image still communicates “close.”

> Rounding the outer window does not automatically round every title-control surface the same way. Work from the window outward: body, title bar, close control and then the content.

Composite widgets have more than one border



Figure 14. The outer inventory frame, grid panel and individual slots use different curves.

Part	Where to work
Containing GUI Window	GUI Window settings.
Widget’s outer frame	The widget’s main Visual Style radius.
Grid panel	Grid Panel radius section.
Repeated cell / save card	Slot Frames radius section.
Save thumbnail area	Save Screenshot radius section.

A Save Game Widget adds a screenshot area inside each card. Inventory item icons are content; they are not the save screenshot frame. Set each layer deliberately instead of changing the root radius until several unrelated shapes happen to look close.

> In each part’s section, -1 inherits that same part’s overall radius. It does not mean “copy every radius from the containing GUI.”

7. Use images as decoration and content

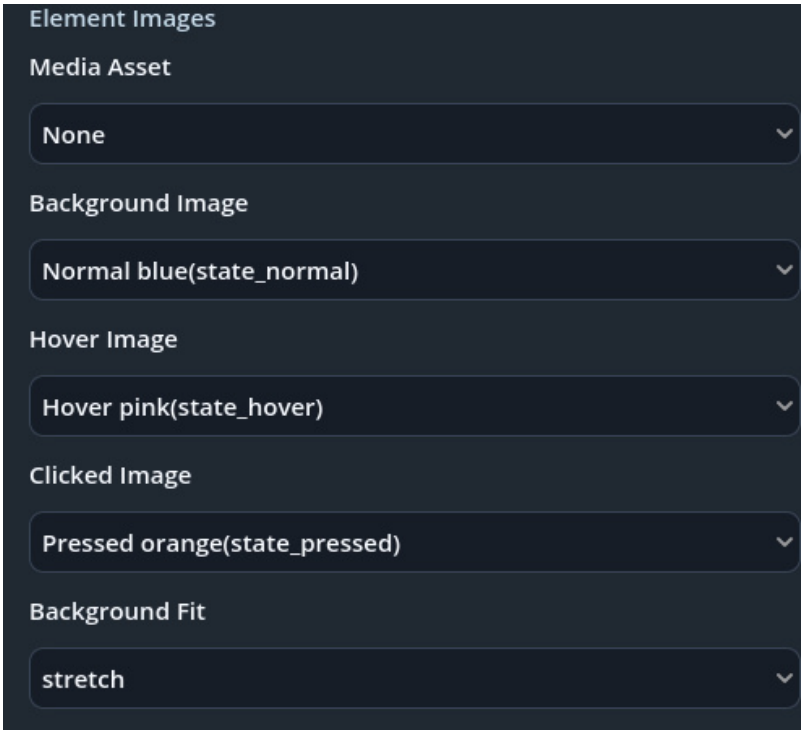

Figure 15. Images separates background artwork from hover and clicked artwork.

Element	Media Asset	Background Image
Button / ordinary backdrop	Can act as a normal backdrop fallback.	Explicit surface artwork behind its caption.
Image	Its main picture content.	A separate backdrop beneath that picture.
Video	The clip.	Backdrop around or behind playback.
3D Model Viewer	The model.	2D backdrop around/behind the rendered model.

Use Background Image for a decorative Button face or panel texture. Use Media Asset for an Image element’s actual illustration. An opaque foreground image can hide the background beneath it.

> When clearing an old Button backdrop, check both Background Image and Media Asset. An empty explicit image can reveal an older fallback rather than remove all artwork.

Fit modes, margins and rounded images

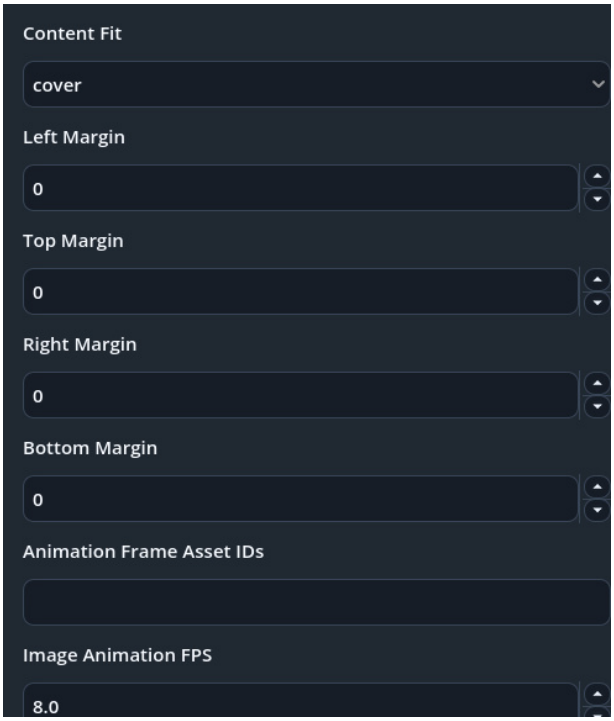

Figure 16. Content Fit and content margins arrange an Image or Video’s content inside its element.

Fit	Useful when…
contain	The whole illustration, logo or portrait must remain visible.
cover	The region must be filled and cropping is acceptable.
stretch	The art is designed to tolerate changed proportions.
tile	A small repeating texture should repeat instead of stretching.

Background Fit belongs to the backdrop. Content Fit belongs to supported media content. Set both intentionally when an element has a picture and a separate background.

Content margins leave room around an Image or Video. They can reveal the backdrop and protect a border. The rounded image renderer clips its image to the authored corner shape; that does not turn every surrounding child or input rectangle into a rounded object.

> A decorative frame image can distort under stretch. The ordinary image fit choices are not a general nine-slice frame editor. Choose art made for the target proportions or use a color/border frame around a contained picture.

Gradients, shadows and animated artwork

A Progress Bar has a built-in two-color fill gradient. Ordinary Button and Panel backgrounds use their authored flat colors and images. A gradient picture is a practical way to give those surfaces a painted transition.

1. For a simple shadow, duplicate or add a plain Panel behind the visible card.
2. Make the lower Panel dark with partial alpha, remove its border if appropriate, and offset it by about 6–8 pixels.
3. Give the shadow a lower Z Index. Keep its curves consistent with the foreground card.
4. Move and test both pieces together. A shadow that stays behind after the card moves is a layout mistake.
5. For a softer or more detailed shadow, import suitable artwork instead of pretending a hard dark rectangle is a blur.

Supported frame lists can animate background artwork. Composite widgets expose Widget Animated Frames, Grid Animated Frames and their FPS fields. Item icon animation is a separate setting in the item definition.

> Use actual frames in an imported asset or a supported frame list. Turning on animation cannot create new frames from a single still picture. Keep animated decoration away from small text.

8. Style feedback without changing behavior

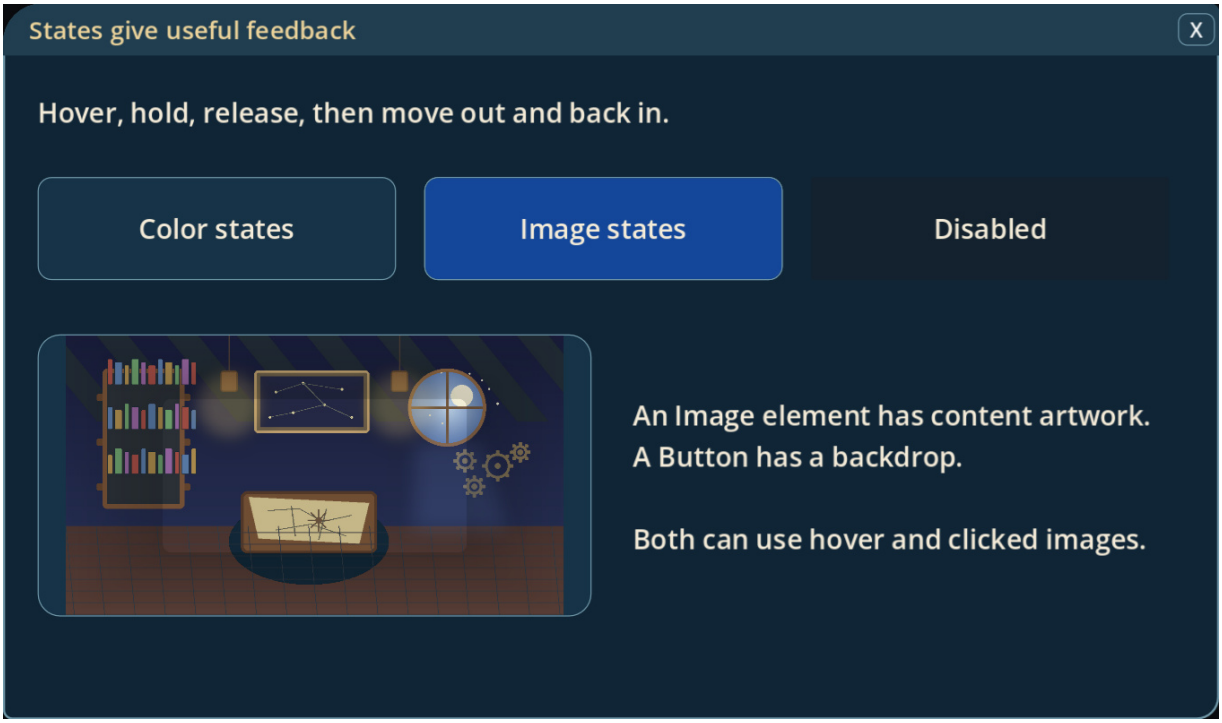

Figure 17. The same menu can show color-based feedback, image-based feedback and a disabled control.

Normal is the resting state. Hover tells the player that the pointer is over a control. Pressed is feedback while an input is held. Selected or checked is a longer-lived choice. Disabled means an operation is unavailable.

Choose feedback the player can recognize quickly: a slightly lighter surface, a firmer outline or a changed image. Do not make a hovered Button look like a different command, or move its art so much that the target seems to jump.

Preview appearance in the designer, then use Play for real input and actions. The designer’s Preview mode lets supported controls respond visually without running the game events that give items, change rooms or save progress.

> A pressed image and an event action are separate. Assigning Clicked Image changes appearance; it does not automatically run a script, and it should not remain as a permanent selected image.

Give press and release a predictable appearance



Figure 18. Held input uses the clicked image; here it is orange.



Figure 19. Release restores normal artwork; here it is blue.

1. Assign a normal background image, a hover image and a clicked image.
2. Enter the Button with the pointer and check hover.
3. Hold the left mouse button or touch and check pressed artwork.
4. Release over the control. The original image should return.
5. Move out and back in to see hover again. Also test a release outside the control.

After release, the image-state system can suppress hover under the stationary pointer until a fresh enter. That prevents the pressed art from appearing to stick. A selected inventory cell or checked option may keep a separate selected style.

> Test keyboard/gamepad focus as well as mouse hover where your game supports it. Keep focus and selected feedback visible; they serve different purposes from a momentary press.

Use the correct state for the correct control

An Image element’s hover and clicked images can replace its content art. On a Button, those image fields provide its surface art behind the caption. Keep the canvas size and composition consistent across each set of state images.

For inventory and saves, image states exist at several regions: widget, grid, slot and save screenshot. A slot’s selected background/border can remain while its pressed picture returns to normal. That is intentional selection feedback.

Checkboxes and radio buttons need a readable checked state. Use explicit wording as well as the mark. A disabled Button should still explain its purpose; avoid reducing everything to an unreadable gray smudge.

Test	Look for
Hover then leave	A useful transition back to normal.
Press then release outside	No stranded pressed picture.
Lose focus during a press	A clean return when input is canceled.
Choose a persistent option	Checked/selected stays distinguishable from held input.
Touch without hover	Every essential action remains discoverable.

If a GUI click also changes its background image through a script, that becomes the new authored/runtime value. Do not confuse an explicit game action with the temporary image-state system.

9. Finish the special controls

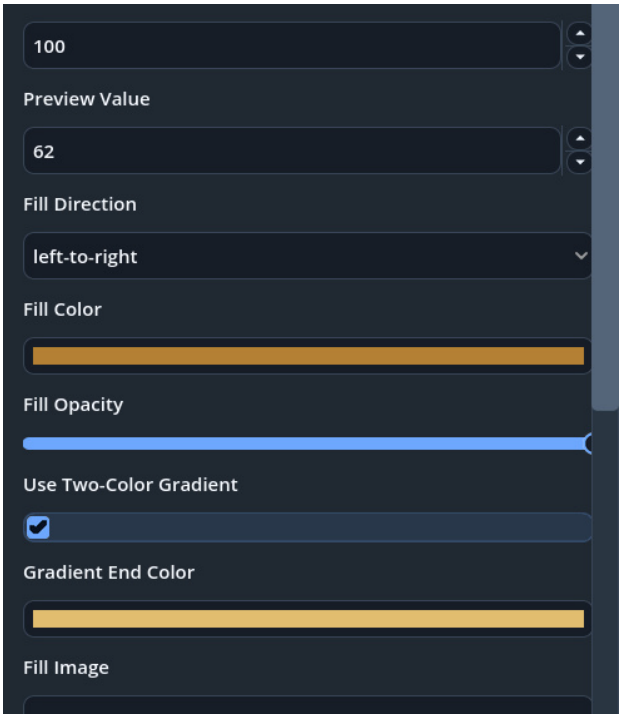

Figure 20. The progress fill has its own color, opacity, gradient and padding controls.

A progress element has a background frame and a separate filled portion. Fill Color, Gradient End Color and Fill Opacity belong to the moving fill. Inner Fill Padding protects the edge. Change the source value separately; styling does not make the bar count time or puzzle steps.

A Slider has a track and a draggable handle. In Images, choose Slider Thumb Image and Slider Thumb Hover Image. Keep the handle large enough to see and grab without covering its neighboring controls.

For List Boxes, inventory, saves, hints and journals, use their type-specific row, entry, slot and navigation controls as well as the outer surface. One global background color is rarely enough to explain all their nested parts.

> A handle image can look larger than its useful drag area. Test the actual interaction, especially on touch hardware, instead of measuring only the painted pixels.

Style media without mistaking it for the room

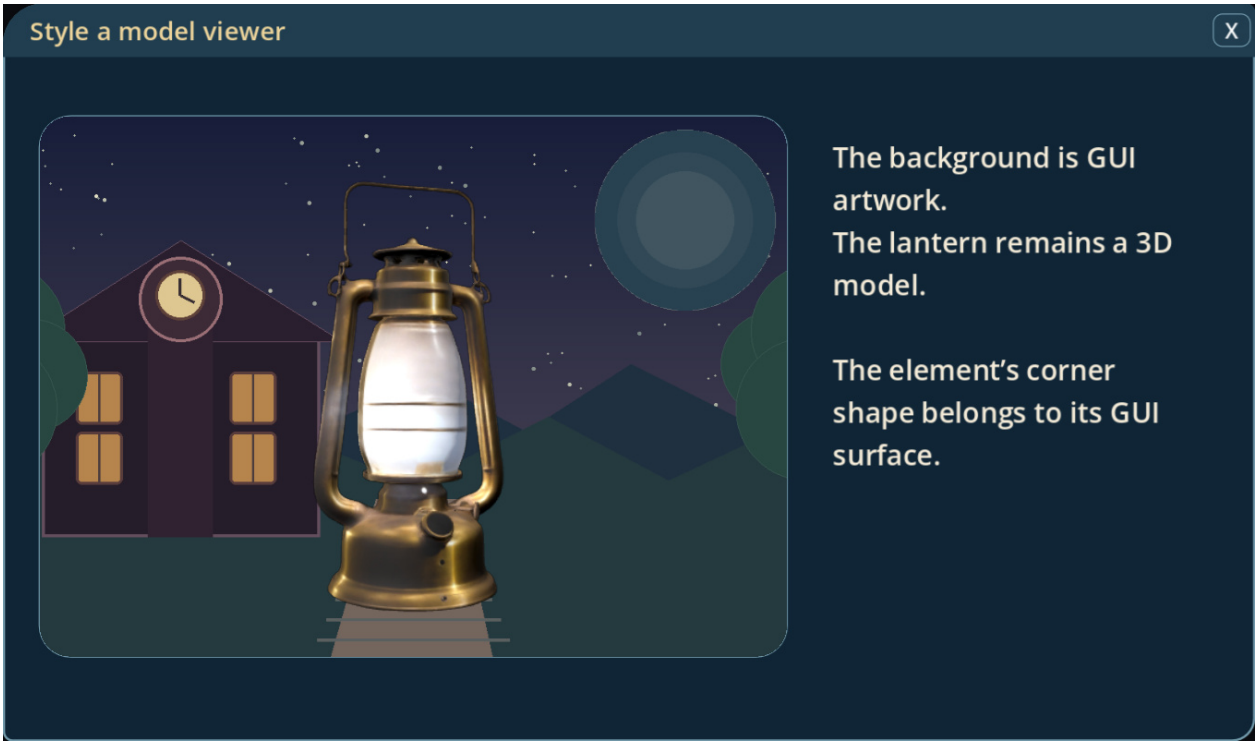

Figure 21. The model viewer uses a 2D GUI backdrop behind a rendered lantern.

For a 3D Model Viewer, select the model through Media Asset and choose its decorative background image separately. Set the viewer’s surface corners, then adjust the model view and fit using its own Type Settings.

A GUI effect processes the viewer’s 2D presentation. It is not the same as changing the model’s physical material, lighting, roughness or scene reflections. Keep those jobs separate when diagnosing the result.

A Video element follows a similar distinction: the clip is content, while background, fit, margins and poster artwork describe its presentation. Avoid strong distortion or heavy grain over a clip that contains important subtitles or clues.

> Room depth of field, scene fog and mirror surfaces belong to the scene rendering systems. They are not a substitute for styling an ordinary GUI control.

10. Add a decorative surface finish

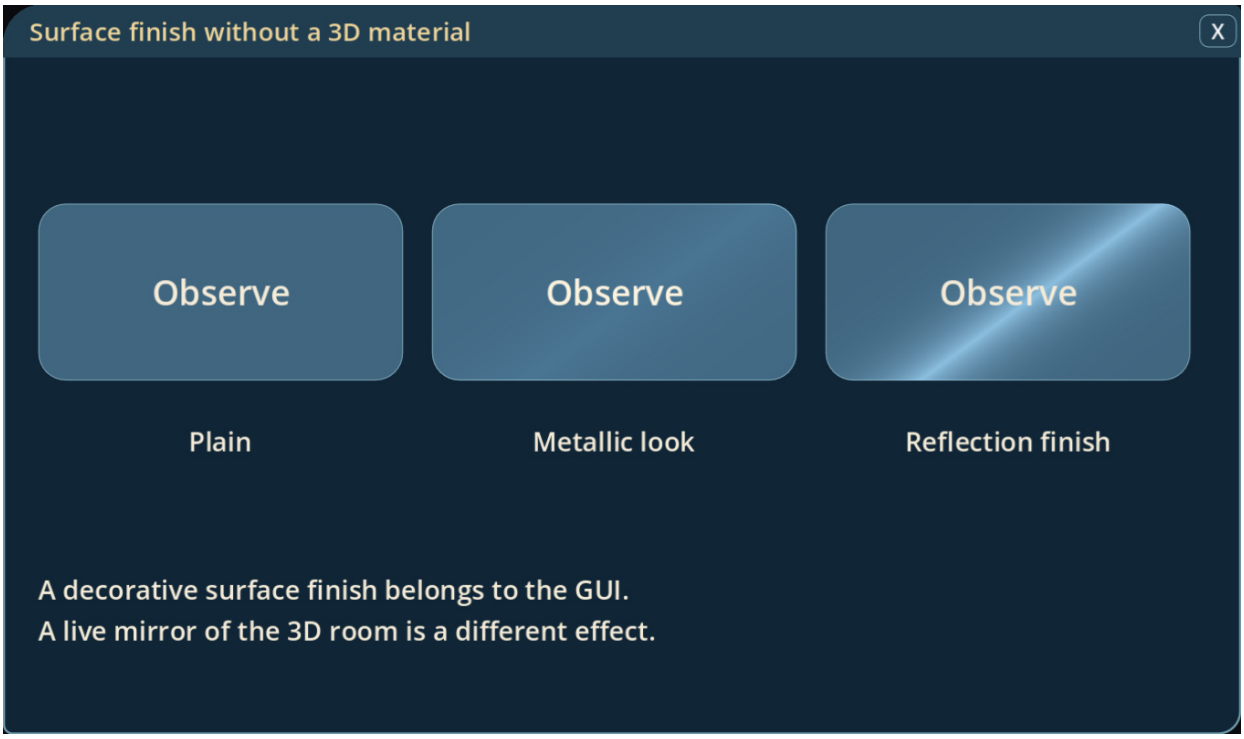

Figure 22. Plain, Metallic Look and Reflection Strength are GUI surface treatments.

Setting	What it contributes
Metallic Look	A stylized brightness/shine treatment.
Reflection Strength	A diagonal reflective-looking band.
Reflection Color	The band’s color; its alpha also affects its contribution.

These controls are useful for a brass instrument panel, polished menu plate or illustrated frame. Start around 0.2–0.4 and compare with zero. The demonstration exaggerates them to make the difference easier to see.

On an ordinary GUI control, Reflection Strength does not create a live mirror of the character or room. The finish is drawn on the 2D surface. A scene mirror needs the scene’s mirror/reflection system instead.

> Finish can affect a control’s internal text and images too. If lettering becomes shiny or washed out, put the decorative plate on a separate element and keep the Label above it.

Keep decorative shine under control

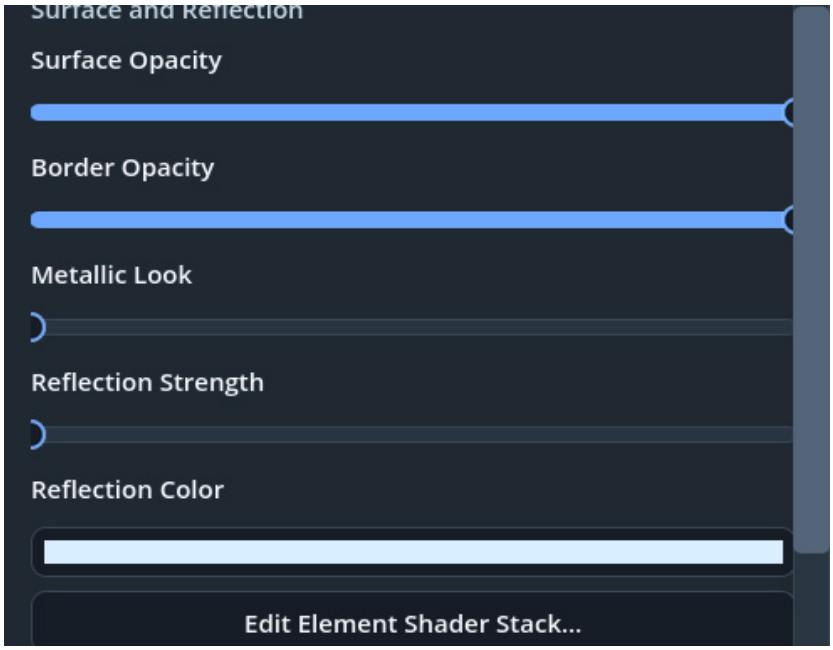

Figure 23. The finish controls live in Visual Style, above the main corner/color fields.

1. Select a Button or decorative Panel.
2. Set Metallic Look and Reflection Strength to zero to establish the plain baseline.
3. Raise one control slowly and check the text and border.
4. Choose a Reflection Color that belongs to the palette, rather than a very bright unrelated color.
5. Test the same control in its normal, hover, pressed and disabled states.

Do not turn several strong finishes and shader effects on at once. It becomes hard to tell whether the washed-out result came from the reflection band, Bloom, a bright hover image or the base color.

> Setting every shader layer to None does not turn Metallic Look or Reflection Strength back to zero. They are separate controls and can keep a material active.

11. Apply your first element shader

The Element Shader tool applies up to eight built-in effect layers to the selected GUI element. For a first test, use an Image with enough color and detail to make the change obvious.

1. Select the Image element. In the example it is Preview Art(preview_art) inside style_lab.
2. Click Element Shader in the toolbar, or Edit Element Shader Stack in Visual Style.
3. Set Layer 1 to Sepia and Layer 1 Intensity to 0.55.
4. Leave the other layers at None and leave Effect Tint transparent.
5. Click Apply to Selected Element(s), then inspect the image in the designer and in Play.

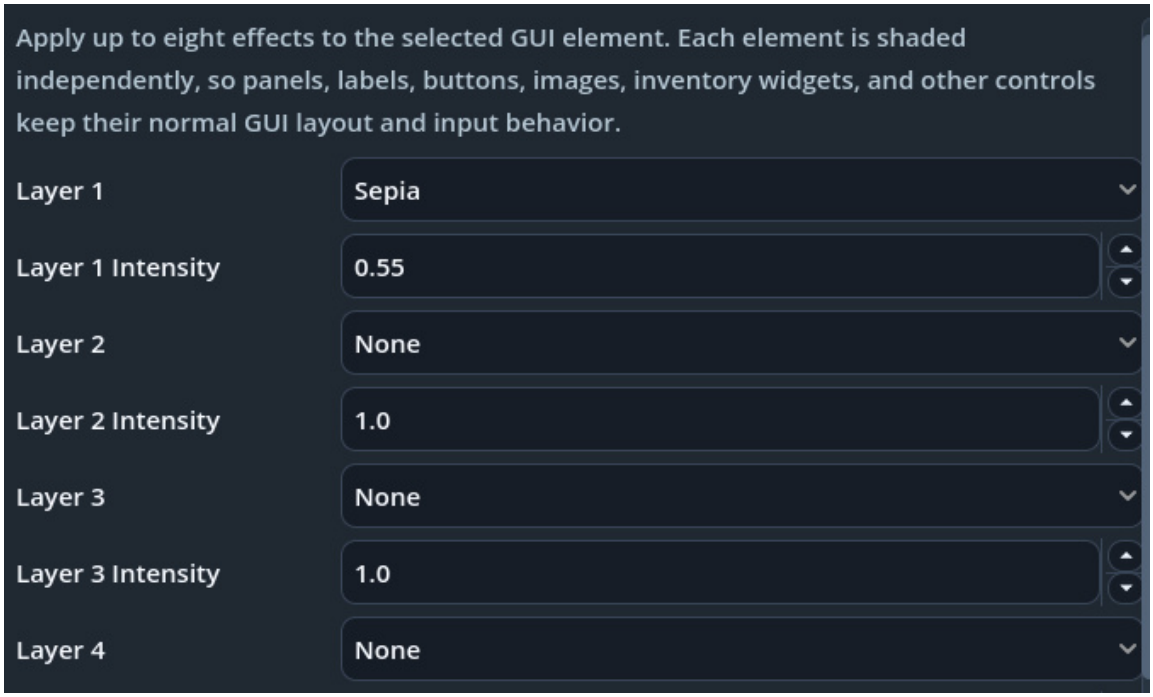

Figure 24. A single moderate layer makes the first test easy to judge.

A shader is not required for a colored Panel or a rounded Button. Establish those ordinary styles first. The effect should add something specific, not compensate for unclear layout.

Judge the result on actual content

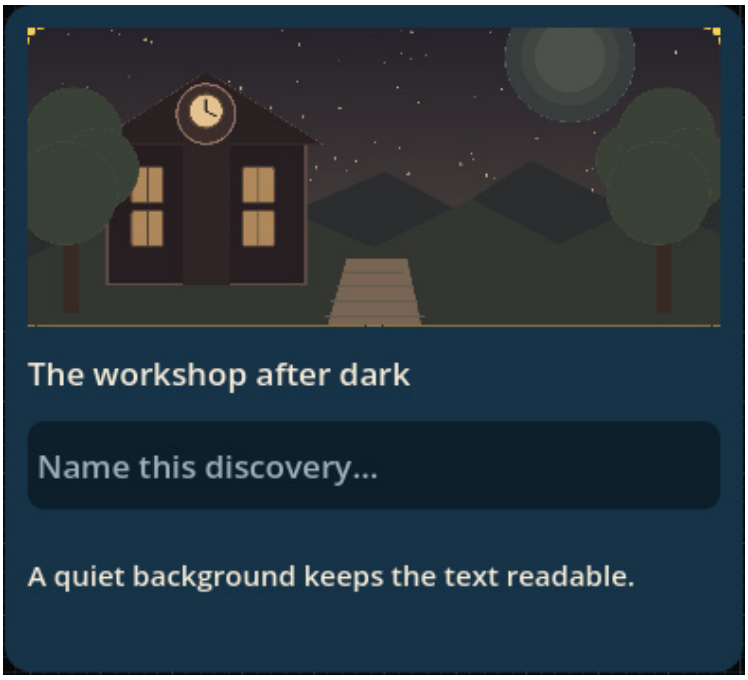

Figure 25. Sepia changes the illustration while separate text below it remains independently styled.

A detailed image is a better first test than a flat-colored rectangle. Pixelization needs detail to break into blocks. Edge effects need contrast changes to find. Grain and tint are easier to judge on midtones than on a solid black area.

Keep one unshaded comparison image or a screenshot of the baseline. Change one layer or parameter, compare it, then move on. The gallery later in this book uses the same image for that reason.

The designer can show the authored effect, but Play is still the final test for animated effects, real input, a model/video’s current frame and your target renderer.

> A paused still screenshot cannot show every aspect of a moving shader. Check motion over several seconds, especially for flicker, grain, VHS effects and distortion.

What the shader actually includes

Selected element	Effect scope
Image	Its rendered image presentation and supported internal surfaces.
Button	Its rendered control tree, including caption/artwork.
Inventory / Save widget	The composite widget’s internal rendered controls as well as its outer surface.
Ordinary Panel	That Panel’s actual control tree; nearby independent elements are not automatically one combined picture.
Several selected elements	Each receives the applied stack separately.

The normal element stack processes rendered 2D surfaces. It does not move the element’s authored position, resize its click area, or turn its corners into a physical 3D mesh.

A separate Label above a decorative Image is a useful arrangement: put the effect on the picture and leave the words clean. This is especially important for puzzle instructions, dialogue choices and save names.

A Panel that looks like it contains a Button may only be visually behind it. Visual overlap, grouping for editing and actual runtime child controls are different relationships. Check the scope with a mild obvious effect before applying a strong stack.

12. Understand the shader controls

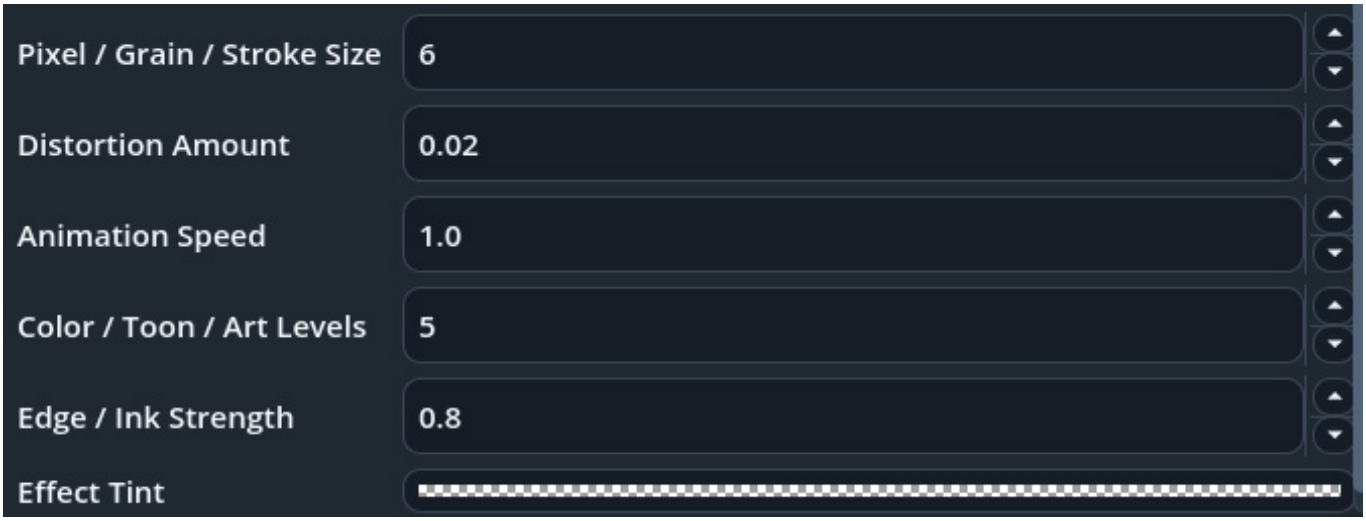

Figure 26. The lower part of the dialog contains shared parameters for the selected stack.

Control	What to try first
Layer Intensity	Start near 0.25–0.65 for a visible but moderate effect. The range is 0–3.
Pixel / Grain / Stroke Size	Begin around 3–6; increase only when the intended pattern is too fine.
Distortion Amount	Begin near 0.005–0.02 for a gentle warp.
Animation Speed	Begin near 0.5–1 for effects that use this parameter.
Color / Toon / Art Levels	Try 4–8 for a simplified-color look.
Edge / Ink Strength	Raise gently while watching fine lines and lettering.
Effect Tint	Leave alpha at zero until you deliberately want color multiplication.

These are shared stack parameters, not independent copies for each layer. Changing Pixel Size can alter several compatible layers in the same stack.

Intensity is not a universal volume knob

Intensity has an effect-specific meaning. For a color effect it often blends toward the treatment; for a distortion it can increase the warp; for a pattern it can change the amount or size. An intensity of 1 is not a guarantee of the same perceived strength across every preset.

If the result is…	Try
Too harsh	Lower that layer’s intensity first.
Too blocky	Reduce Pixel Size as well as checking intensity.
Too few colors	Raise Color Levels or lower the simplification layer’s intensity.
Edges too dark or busy	Lower Edge / Ink Strength and the outline layer’s intensity.
Moving too much	Reduce the appropriate speed/amount, or choose a static treatment.

Not every effect uses every shared parameter. Do not expect Color Levels to change a simple warm tint, or Reflection Color to change the grain pattern. The gallery tables point to the useful controls for each family.

> For a truly still alternative, remove the animated preset. Animation Speed 0 freezes effects that use it directly, but some effects retain time-based motion or a minimum update rate. It is not a universal reduced-motion switch.

Effect Tint multiplies the rendered color

Effect Tint uses RGB for the tint color and alpha for how strongly it is mixed in. With alpha zero it contributes nothing. With a stronger alpha, the existing color is multiplied toward the tint’s RGB values.

Tint choice	Likely result
White, alpha 0	No tint.
White, alpha 1	Multiplication by white leaves the color largely unchanged.
Pale blue, partial alpha	A cooler cast.
Dark brown, high alpha	A substantial darkening as well as a warm cast.
Black, high alpha	The image can become nearly black.

Use a pale tint with moderate alpha before trying a dark saturated one. Read the picture’s darkest details and the control’s text after applying it.

In this build, a tint by itself with all layers None and no surface finish may not activate a GUI effect material. Start with an active layer, such as a mild Warm or Cool treatment, before judging Effect Tint.

Effect Tint is a stack-wide parameter. Text Color is the ordinary caption color. Reflection Color belongs to the surface finish. Three color pickers can legitimately change three different things.

13. Pick an effect for a reason

The following pages cover every built-in choice in the GUI Element Shader picker for this reference build. Most comparisons use intensity 0.65, Pixel Size 4, Color Levels 5, Edge Strength 0.65, Distortion Amount 0.018 and Speed 0.7.

Desired result	Start with
Old paper or memory	Sepia, Warm, Film Grain.
Terminal or projection	Hologram, CRT, Cool.
Simplified artwork	Posterized, Cel Shaded, Toon.
Handmade illustration	Color Sketch, Color Crayon, Hand Drawn.
Signal damage or instability	VHS CRT 2, Screen Flicker, Distorted.
A bright focal illustration	Bloom or a modest Photography treatment.

These are starting points. Keep captions on separate unshaded elements when the treatment interferes with reading. A setting that looks good on a large picture can be a poor choice for a tiny inventory icon.

> PSX, N64 and Horror Fog belong to the engine’s other rendering paths and are not choices in this GUI Element Shader picker. Likewise, a scene mirror or depth-of-field blur is not an ordinary GUI preset.

Color, age and simplified shading

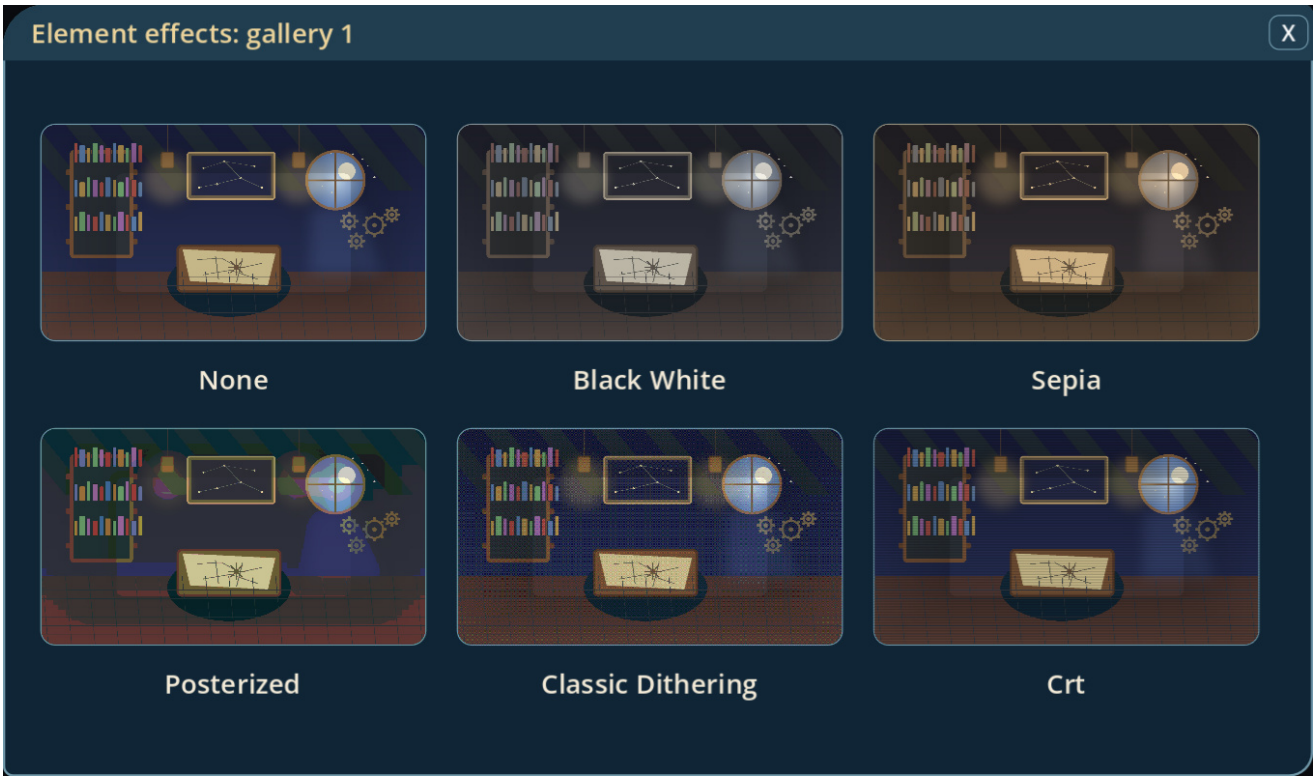

Figure 27. Original, Black White, Sepia, Posterized, Classic Dithering and CRT.

Effect	Useful purpose / control
None	The unprocessed comparison.
Black White	A monochrome photo, disabled illustration or memory; intensity controls the blend.
Sepia	A warm aged image; protect small dark details.
Posterized	A limited-color treatment; Color Levels matters.
Classic Dithering	An ordered color pattern; compare it at the actual game size.
CRT	Scanline-style display treatment; avoid applying it to small text by default.

Warmth, brightness and signal texture



Figure 28. Warm, Cool, Bloom, Photography, Film Grain and Hologram.

Effect	Useful purpose / caution
Warm / Cool	A modest color cast; choose one consistent direction for related artwork.
Bloom	Brightens and softens bright image features. It is a GUI image treatment, not a 3D light source.
Photography	A gentle photographic tonal treatment.
Film Grain	Texture and movement; inspect flat areas and small lettering.
Hologram	A cyan projection/scanline look; keep the main instructions separate.

GUI Bloom does not enlarge the control’s geometry or guarantee an external halo outside its drawn shape. Use suitable artwork when you need a large painted glow around a frame.

Illustration, ink and faceted looks

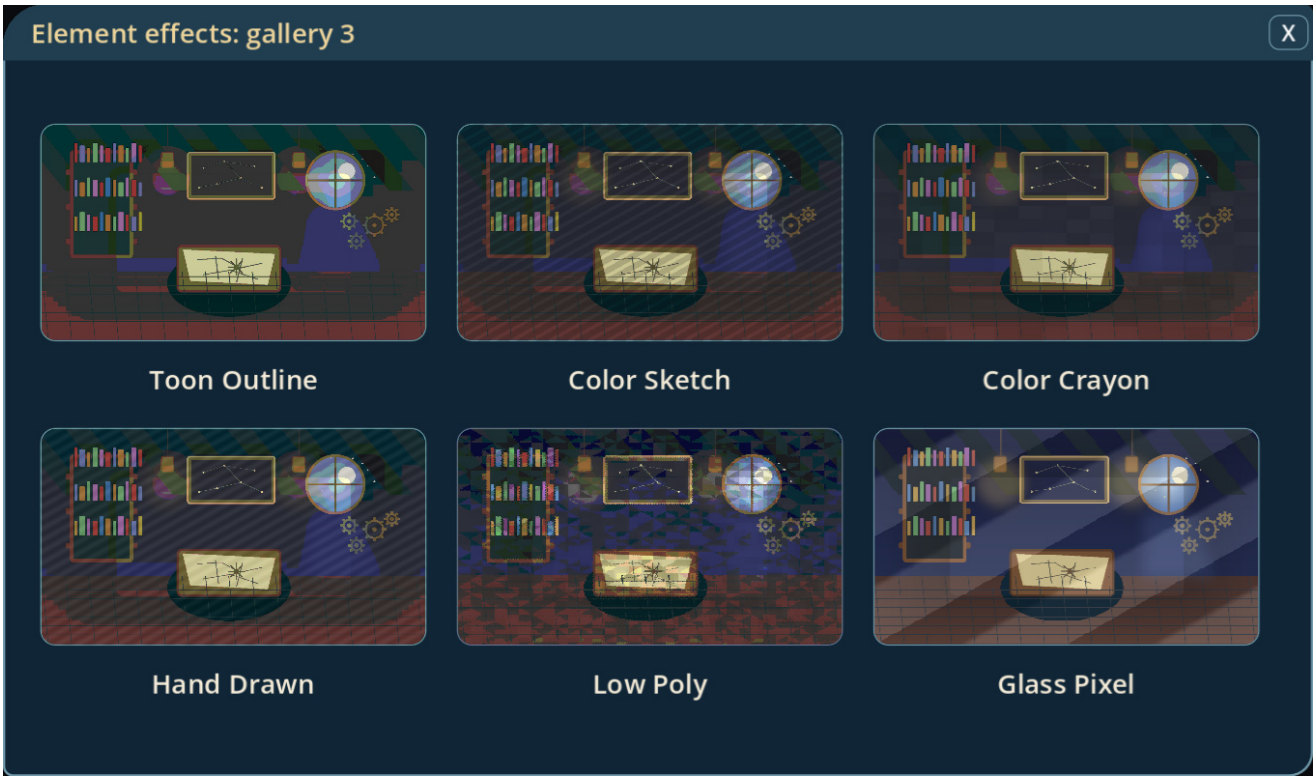

Figure 29. Toon Outline, Color Sketch, Color Crayon, Hand Drawn, Low Poly and Glass Pixel.

Effect	Useful purpose / control
Toon Outline	Simplified image plus dark edge treatment; Edge / Ink Strength is important.
Color Sketch / Color Crayon	Illustrative color and texture; compare Color Levels and pattern size.
Hand Drawn	A moving drawn treatment; check whether motion suits the menu.
Low Poly	A faceted-looking 2D treatment. It does not reduce a model’s polygon count.
Glass Pixel	Pixel blocks with a shine treatment; it is not a live blur of the room behind the panel.

Distortion, frost and dramatic surfaces

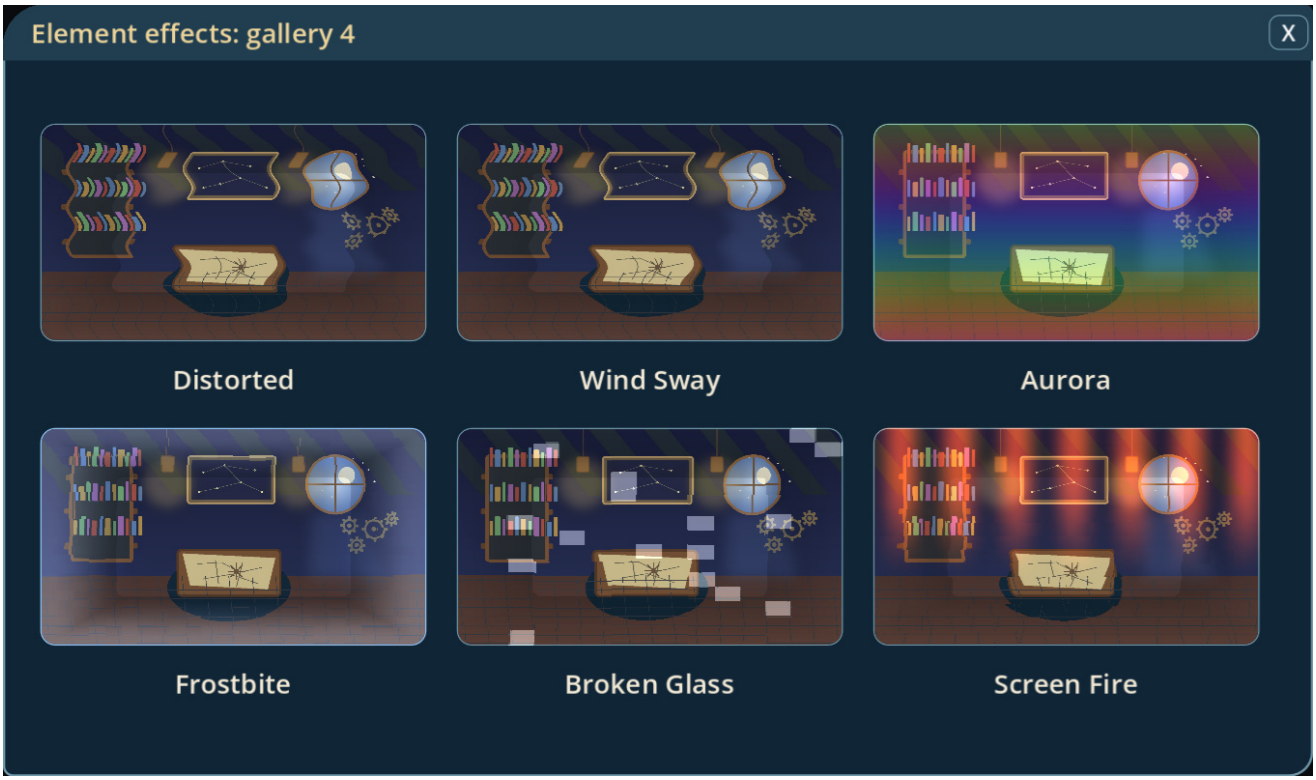

Figure 30. Distorted, Wind Sway, Aurora, Frostbite, Broken Glass and Screen Fire.

Effect	Useful purpose / caution
Distorted / Wind Sway	Warp the sampled image. Start with a small Distortion Amount.
Aurora	An animated color treatment for a decorative image or magical display.
Frostbite	Cold edge/warp treatment; test the border region.
Broken Glass	Segmented image offsets; it does not create physical glass shards.
Screen Fire	Animated heat/fire-like color and warp. It does not create a scene fire emitter or provide lighting.

Use dramatic treatments on decoration. Essential instructions, save names and confirmation buttons should remain easy to recognize.

Heat, damaged displays and metallic art



Figure 31. Heat Haze, VHS CRT 2, Screen Flicker, Metallic Toon, Invert and Vignette.

Effect	Useful purpose / caution
Heat Haze	A special background-refraction path; use it alone and read the next page.
VHS CRT 2	A noisy, displaced display image; Amount and Speed matter.
Screen Flicker	Visible brightness changes over time; provide a calmer alternative.
Metallic Toon	Stylized metallic image colors, separate from the ordinary Metallic Look slider.
Invert	Negative-like colors; use sparingly for a specific state or story beat.
Vignette	Darkens image edges; important clues near the edge can become hard to see.

Pixelized, Cel Shaded and Toon



Figure 32. The final picker choices simplify the source image in different ways.

Effect	What to adjust
Pixelized	Pixel / Grain / Stroke Size and intensity determine the visible block treatment.
Cel Shaded	Color / Toon / Art Levels changes the simplified shading.
Toon	A cartoon-style tonal treatment; compare it with Cel Shaded on your actual art.

For pixel art, begin with a source that already has a deliberate pixel grid. Applying a block shader to a high-resolution menu is an appearance treatment, not a complete pixel-art conversion workflow.

A small icon may lose its identifying feature under heavy simplification. Check the actual key, notebook or portrait at the size players will see, rather than judging only a large gallery illustration.

> None is the right choice when the source artwork already has the intended style. A clean icon does not need a shader simply because the rest of the menu uses one.

Heat Haze is a special case

Heat Haze samples the background like rising hot air. On a local GUI surface, it can replace the surface’s own picture with a softened-edged refracted view behind it. That is why the Heat Haze gallery tile does not behave like a simple color filter over the map.

1. Use a small decorative image/surface with no essential text.
2. Choose Heat Haze as the only effect layer.
3. Start with Intensity 1, Distortion Amount 0.02 and Animation Speed 1.
4. Put a visible background behind it and run the game.
5. Use Amount 0 to remove distortion and Speed 0 to freeze this haze’s motion while comparing it.

Heat Haze has its own material path. Do not assume other ordinary stack layers or the normal tint/finish parameters combine with it. Use it alone for a predictable local plume.

It can be a useful effect over a magical instrument or a hot interface illustration, but it is rarely a sensible treatment for an entire save browser. The player needs to read that browser without looking through a mirage.

> Screen Fire adds a fire-like image treatment. Heat Haze bends the view without adding ordinary flames or tint. Choose according to the visual job you actually need.

14. Build a stack one layer at a time

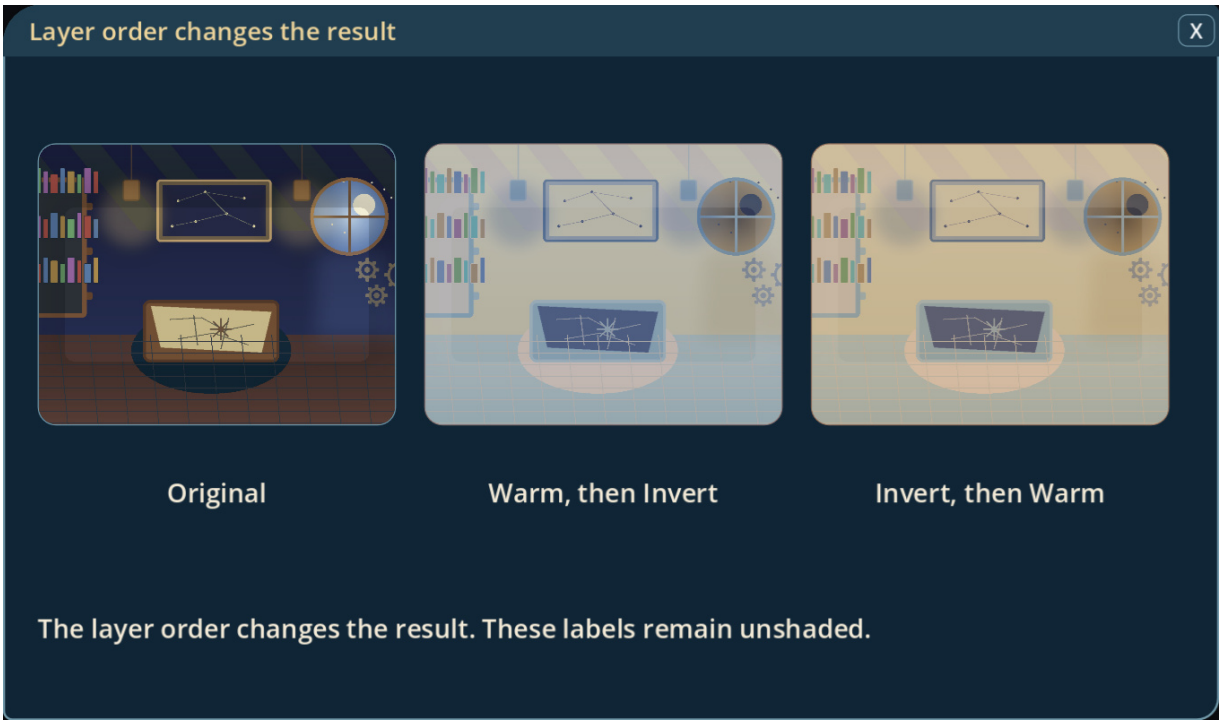

Figure 33. Warm then Invert and Invert then Warm produce different colors from the same source.

1. Begin with all layers None and finish strengths at zero.
2. Add the most important effect in Layer 1. Compare against the baseline.
3. Add a second effect only when it has a clear purpose.
4. Compare both orders. Lower intensity before adding another layer.
5. Check shared parameters again; they can affect more than one layer.

The stack is ordered, but it is not a miniature paint program with independent masks and blend modes for every row. The runtime uses the layer list for supported image-coordinate and color-processing passes. Judge mixed warp/color combinations visually rather than assuming every possible operation is flattened in a simple painted sequence.

> Leave unused layers at None. Eight available slots are capacity, not a target.

Apply to several elements carefully

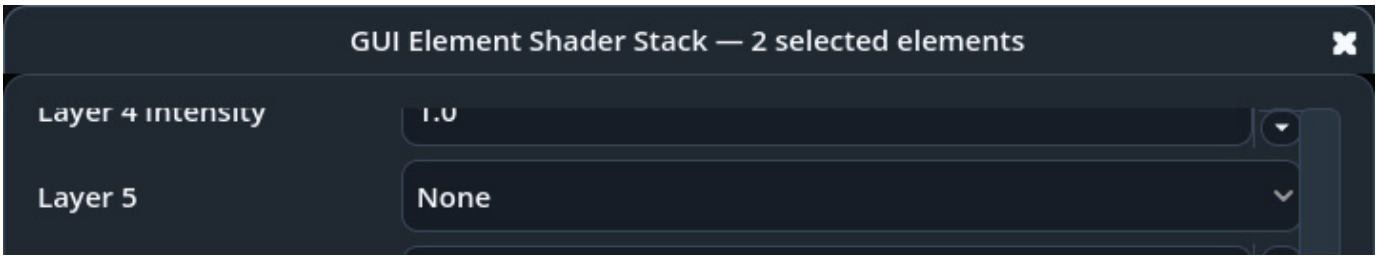

Figure 34. The dialog identifies a multiple-element selection before applying the stack.

Ctrl-click the intended elements in the GUI Elements list, then open Element Shader. The dialog starts from the primary selected element’s settings and applies the resulting stack to each selected element.

This is useful for a set of matching decorative images. It can also overwrite stacks that were deliberately different, so inspect the selection before pressing Apply. Cloning a finished element is often a clearer way to create a coordinated set.

The elements are processed separately. A set of Buttons is not automatically flattened into one shared screen-sized texture. Their individual borders, captions and image sizes still matter.

> Do not include an important Label in a mass shader change merely because it sits near the artwork. Select the decoration explicitly and keep the reading layer separate.

Remove effects and get back to neutral

To remove…	Reset
Built-in element layers	Every layer to None, then Apply.
Decorative shine	Metallic Look and Reflection Strength to 0.
Tint contribution	Effect Tint alpha to 0, while checking whether any active material remains.
Temporary state artwork	Clear the relevant hover/clicked image assignment.
An ordinary image backdrop	Clear Background Image and inspect the Media Asset fallback.
An old inherited stack	Give the element its own explicit empty shader object or apply an all-None stack.

Older projects can have a GUI-level shader used as a fallback for elements without their own shader field. An explicit empty object, "shader": {}, opts that element out of the legacy default. An all-None apply also records an explicit element stack.

Imported shader assets do not automatically become new entries in the built-in Element Shader picker. The workflow in this book uses the supported presets. A custom canvas shader integration needs its own explicit supported assignment or code path.

> A completely neutral test means more than None in Layer 1. Check later layers, surface finish, tint, image states and ordinary base colors too.

15. Give the window an entrance and exit

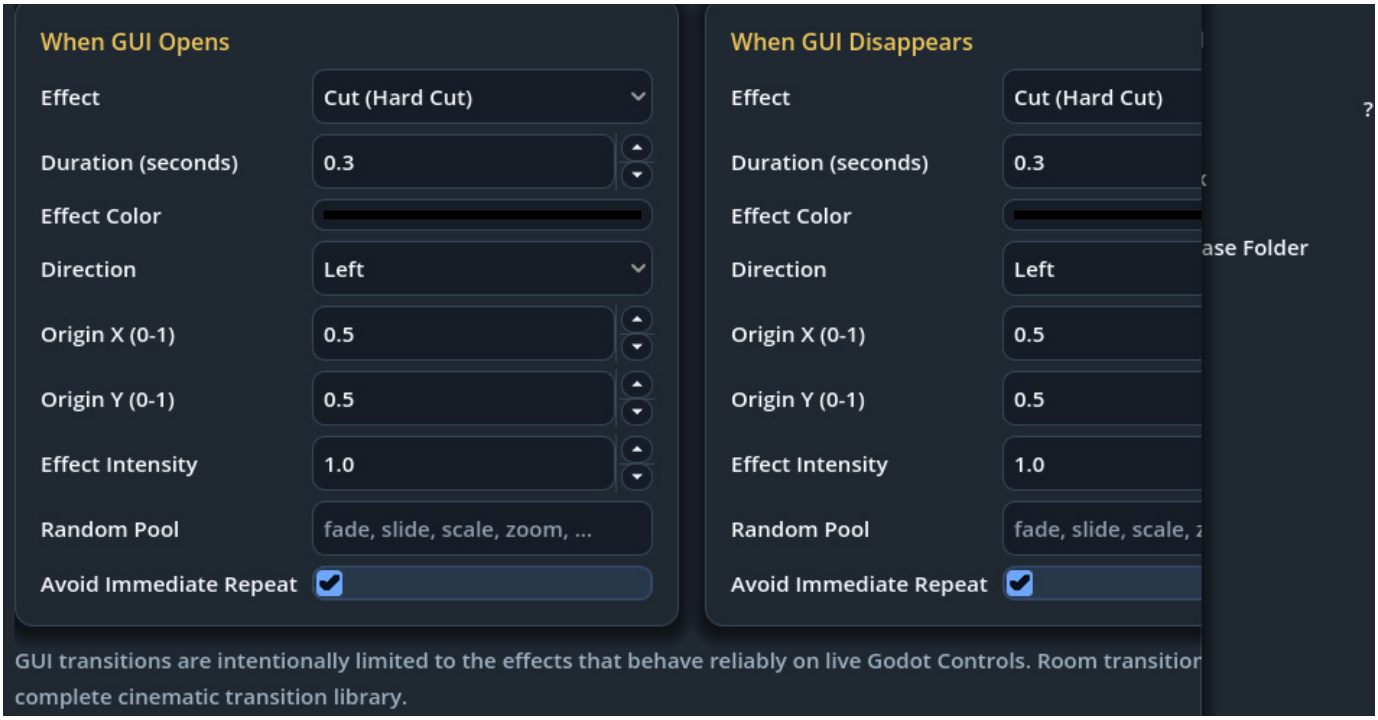

Figure 35. Open and disappear transitions are authored separately for each GUI.

1. Choose the GUI, then click Transitions in its toolbar.
2. Set When GUI Opens to Fade. Try 0.25–0.4 seconds for an ordinary menu.
3. Set When GUI Disappears to Slide, or keep it equally restrained with Fade.
4. Review Direction, Origin and Intensity where the chosen effect uses them.
5. Click Apply Transitions. In Play, hide and show the GUI to test the full cycle.

A transition changes the window’s temporary presentation. An element shader describes how a control looks while it is present. They can be used together, but they are different systems.

> The screenshots use longer 1.2-second transitions so a middle frame can be captured. Most everyday menus should open much more quickly.

Choose a reliable GUI transition

GUI transition ID	Use
cut	Immediate appearance or disappearance.
fade	A restrained opacity transition.
slide	Movement from/to a direction.
scale / zoom	A size-based entrance or exit; judge the origin.
character-blink	A stylized blink presentation.
grid-checkerboard	A more patterned reveal.
comic-panel	A more theatrical panel-like presentation.
random	Choose from the authored pool of reliable GUI effects.

The GUI list is deliberately smaller than the room’s cinematic transition library. Do not copy every room transition name into a GUI pool and assume it is supported.

Direction and Origin help define movement/scale effects. The origin uses 0–1 coordinates; 0.5, 0.5 is the center. Color and Intensity matter only where the chosen effect uses them.

For Random Pool, enter comma-separated supported IDs. An empty pool allows the reliable GUI choices. Avoid Immediate Repeat reduces the chance of the exact same random effect happening twice in succession.

> Predictability usually suits an inventory, options screen or save menu. Reserve random transitions for an interface whose fiction benefits from that variation.

Test the middle of the animation too

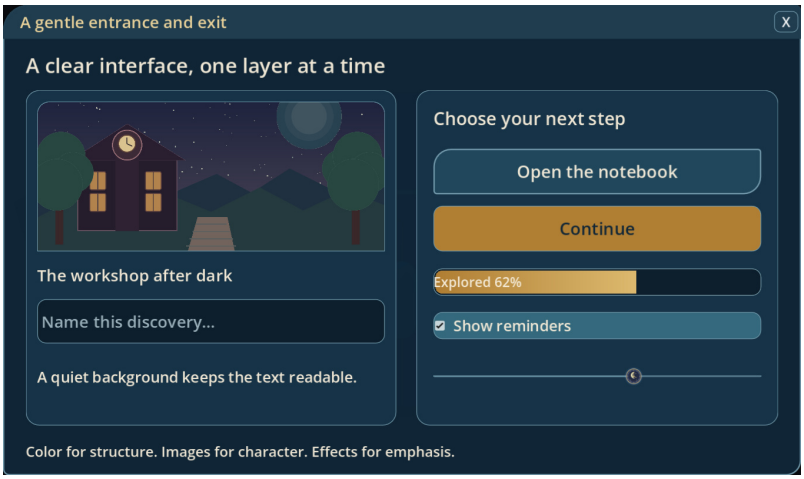

Figure 36. A partially opened window during the deliberately slow fade.

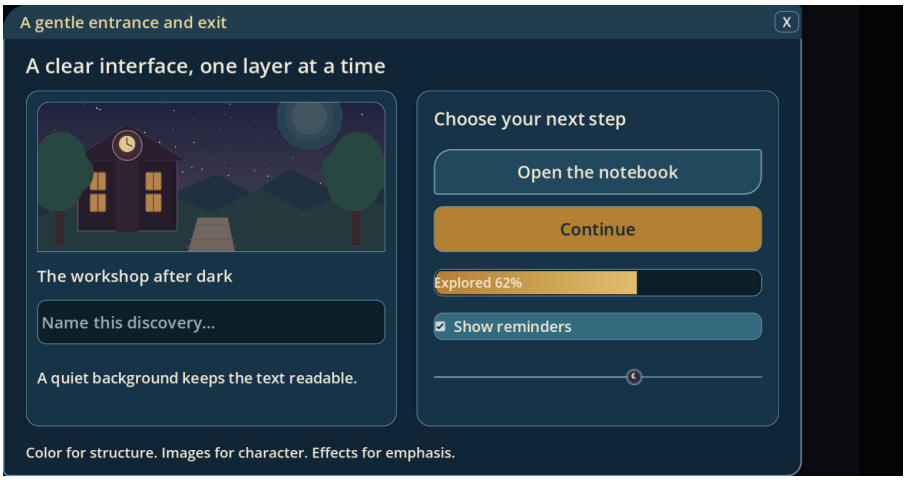

Figure 37. A middle frame of the closing slide.

Open, close and reopen the menu quickly. Check that it returns to its intended position and size, that controls are usable once settled, and that the game is not left in an unwanted input state.

Test long captions, clipped window contents, a model/video element and any mouse-position activation you use. A pleasant animation that leaves the save button in the wrong place is not finished.

> Provide a simple Cut or short Fade alternative when the audience or platform benefits from reduced motion. Changing a transition should preserve the same menu behavior.

16. Change styling during play

The ordinary shared actions can change GUI and element properties during gameplay. This is useful for a selected tab, an alert, a story-driven palette change or a player preference.

1. Add a visual script or action list for the change.
2. Choose Change GUI Element Property.
3. Select GUI style_lab and Element primary_button.
4. Choose property radiusTopLeft and set To to 32.
5. Run the action during Play and inspect that corner.

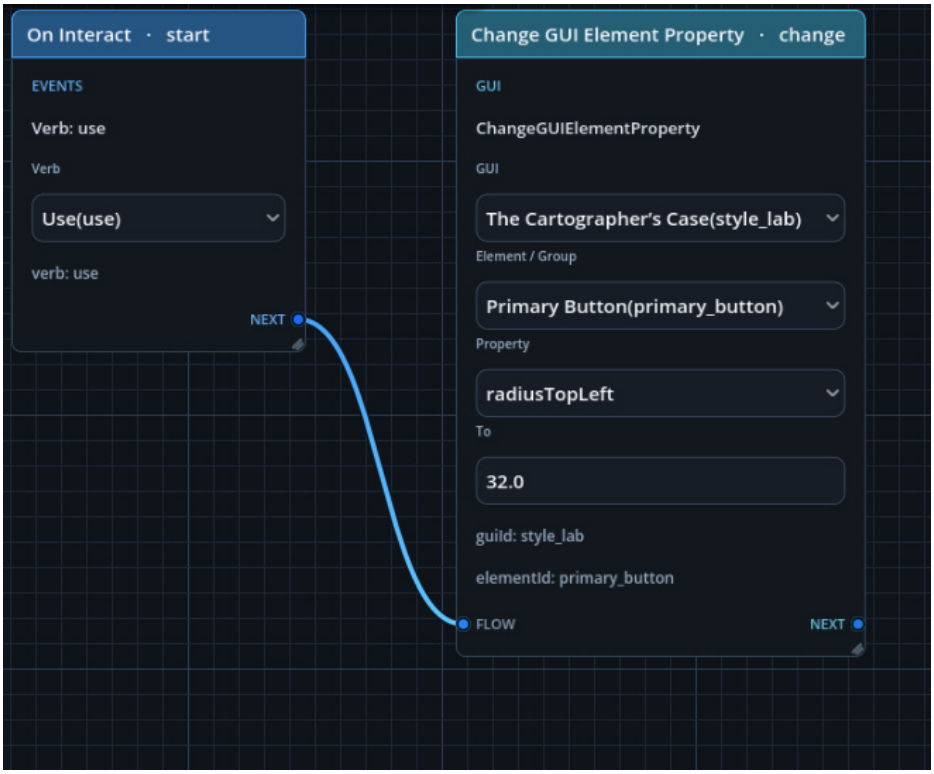

Figure 38. A visual node changes one corner on one identified element.

> Changing the common radius does not erase explicit per-corner overrides. If Top Left is already 24, setting radius to 32 does not replace that 24. Change radiusTopLeft or reset it to -1.

Use a script to change a complete state

Changing only the normal background can leave hover and pressed colors in the old palette. This complete GDScript record updates the related colors together:

```
extends "res://scripts/runtime/script_base.gd"

func run(_context: Dictionary = {}) -> Variant:
    var updates: Dictionary = {
        "background": "#713e35",
        "hoverBackground": "#8d5946",
        "pressedBackground": "#542a24",
        "borderColor": "#efc577"
    }
    for property_name: String in updates:
        await ChangeGUIElementProperty({
            "guiId": "style_lab",
            "elementId": "primary_button",
            "property": property_name,
            "value": updates[property_name]
        })
    return true
```

Create a normal GDScript record, apply it, then assign its ID to the intended script-running event or action. Replace the GUI and element IDs with your own. Do not paste the function into a GUI’s raw element JSON.

Use the same idea for a normal/selected pair of tab styles. Put the change at the point where selection changes, rather than repeatedly rebuilding a whole menu every frame.

> These are runtime changes. They do not mean the editor project has been permanently redesigned. Store an important theme choice in supported game settings/variables, reapply it deliberately and test it after loading a save.

Change a shader as one complete property

```
extends "res://scripts/runtime/script_base.gd"

func run(_context: Dictionary = {}) -> Variant:
    var treatment: Dictionary = {
        "preset": "none",
        "layers": [{"preset": "warm", "intensity": 0.45}],
        "pixelSize": 4,
        "colorLevels": 5,
        "edgeStrength": 0.65,
        "distortionAmount": 0.01,
        "distortionSpeed": 0.7,
        "tintColor": "#ffffff00"
    }
    await ChangeGUIElementProperty({
        "guiId": "style_lab",
        "elementId": "preview_art",
        "property": "shader",
        "value": treatment
    })
    return true
```

The shader is a Dictionary containing the layer list and shared parameters. Replace the shader property with that complete object. Do not assume a dotted key such as shader.layers.0.intensity is a general deep-update command in every GUI action route.

The example applies a mild Warm effect to the illustration while leaving the separate Label clean. The same shared action is available to action lists and visual scripting when the value is supplied as a structured Dictionary.

If the property selector is built from the current element, apply an element shader once in the editor so shader is an authored field. The GDScript call above supplies its exact key directly.

Return an element to a plain surface

```
extends "res://scripts/runtime/script_base.gd"

func run(_context: Dictionary = {}) -> Variant:
    var updates: Dictionary = {
        "shader": {},
        "metallicStrength": 0.0,
        "reflectionStrength": 0.0
    }
    for property_name: String in updates:
        await ChangeGUIElementProperty({
            "guiId": "style_lab",
            "elementId": "preview_art",
            "property": property_name,
            "value": updates[property_name]
        })
    return true
```

The explicit empty shader opts out of a legacy GUI-level fallback. The two finish values remove the separate shine controls. This does not clear ordinary background colors, borders, content images or hover-image assignments.

Use this as the start of a reduced-effects option, then also review moving GUI transitions and any other animated artwork. Removing a shader does not stop a video clip or an animated icon.

> Keep a simple reset path while developing a heavily styled screen. It is much easier to find the cause of a problem when you can compare a known plain state.

Change the containing window

```
extends "res://scripts/runtime/script_base.gd"

func run(_context: Dictionary = {}) -> Variant:
    await ChangeGUIProperty({
        "guiId": "style_lab",
        "property": "windowFrame.windowBackground",
        "value": "#182b39"
    })
    await ChangeGUIProperty({
        "guiId": "style_lab",
        "property": "windowFrame.radiusTopLeft",
        "value": 0
    })
    return true
```

Change GUI Property addresses the window. Change GUI Element Property addresses a particular element. The explicit windowFrame prefix in this script identifies the stored window-chrome property.

A project can have several GUIs open, each with a different background. Supply the GUI ID deliberately so an alert style changes the intended window.

> For a full theme change, update the title bar, close control, element colors, image states and text as appropriate. Changing the outer window alone does not retheme the separate controls inside it.

Use the shared action from other languages

The styling action is shared. These small examples all change the art Panel’s common radius to 24. That Panel’s corners use -1, so they inherit the new value.

PCA Script:

```
language 1.0

let p = {"guiId":"style_lab","elementId":"art_panel","property":"radius","value":24}
change_gui_element_property(p)
```

Lua:

```
local Script = {}
function Script.run(context)
    await(ChangeGUIElementProperty({
        guiId = "style_lab", elementId = "art_panel",
        property = "radius", value = 24
    }))
    return true
end
return Script
```

BASIC:

```
OPTION PCA3D BASIC 1.0
FUNCTION Run(context AS Dictionary)
    DIM p = {"guiId":"style_lab","elementId":"art_panel","property":"radius","value":24}
    AWAIT ChangeGUIElementProperty(p)
    RETURN TRUE
END FUNCTION
```

Use numerical values for radii and opacity. Use strings for hex colors and asset IDs. A shader value is a Dictionary, not a string containing a pretend shader name.

Use advanced JSON when a typed field is absent

Some element types expose a dedicated Border Width field, while an ordinary Button or Panel may not show that same typed control in Visual Style. The supported authored borderWidth property can still be edited in Raw JSON.

1. Select the intended element and open Raw JSON (Advanced).
2. Keep the full existing element object. Find borderWidth; change it to 2, or add it as a correctly comma-separated property.
3. Find the radius and corner properties if you are adjusting those at the same time.
4. Click Validate and Apply Element JSON (Advanced).
5. Check the visible border, then Save Project.

These lines are an excerpt to merge into the existing object, not a replacement for the whole element:

```
"borderWidth": 2,
"radius": 16,
"radiusTopLeft": 24,
"radiusTopRight": 4,
"radiusBottomRight": 24,
"radiusBottomLeft": 4
```

JSON uses double quotes around keys and strings. Numeric values are not quoted. Separate properties with commas and avoid a trailing comma after the last property in an object.

> Do not replace the entire element with the excerpt. Its ID, type, position, size, content and events must remain. Save a project copy before large raw edits.

17. Build three complete looks

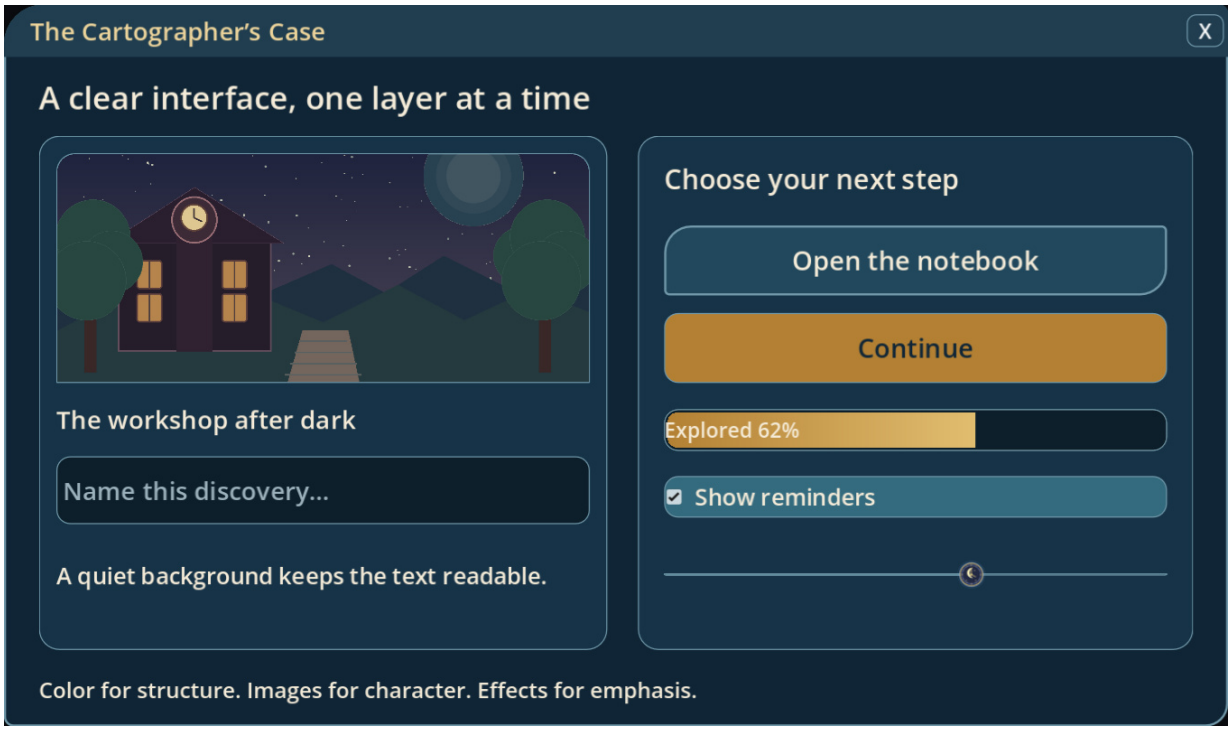

Figure 39. The midnight-and-brass recipe uses contrast and one warm accent.

Part	Midnight and brass
Window / Panel	#102637 / #173347
Text / border	#F5ECD9 / #7199AA
Button / hover / pressed	#21485D / #356B7F / #825F3A
Primary accent	#B48134 with dark text
Corners	Window 20; Panels 18; a deliberate asymmetric Button.
Effect choice	Leave most text plain; a mild Warm or Photography layer on an illustration.

Use the warm accent for the most important choice or meter. Keep secondary controls in the blue family. The illustration supplies most of the visual detail, so the controls do not need strong grain or distortion.

> Clone this finished GUI for related menus, then give the clone appropriate IDs and update its event targets. A copied appearance should not accidentally keep a command aimed at the old menu.

A parchment interface



Figure 40. The same layout with a light surface, dark ink and brown accents.

Part	Parchment recipe
Window / Panel	#E7D7AF / #F3E7C7
Main text	#332B23
Border	#766346
Button accent	#6D4B2C
Button caption	#FFF5DD
Optional image treatment	Mild Sepia; keep small captions unprocessed.

Changing from dark to light requires more than swapping the window background. Recheck placeholder text, progress captions, the close control, selected rows and disabled choices. A pale caption that worked on midnight blue can disappear on parchment.

> A painted paper texture can add character, but avoid heavy markings directly under long descriptions or save names. Use the light plain panel as a reading area.

A cool terminal interface

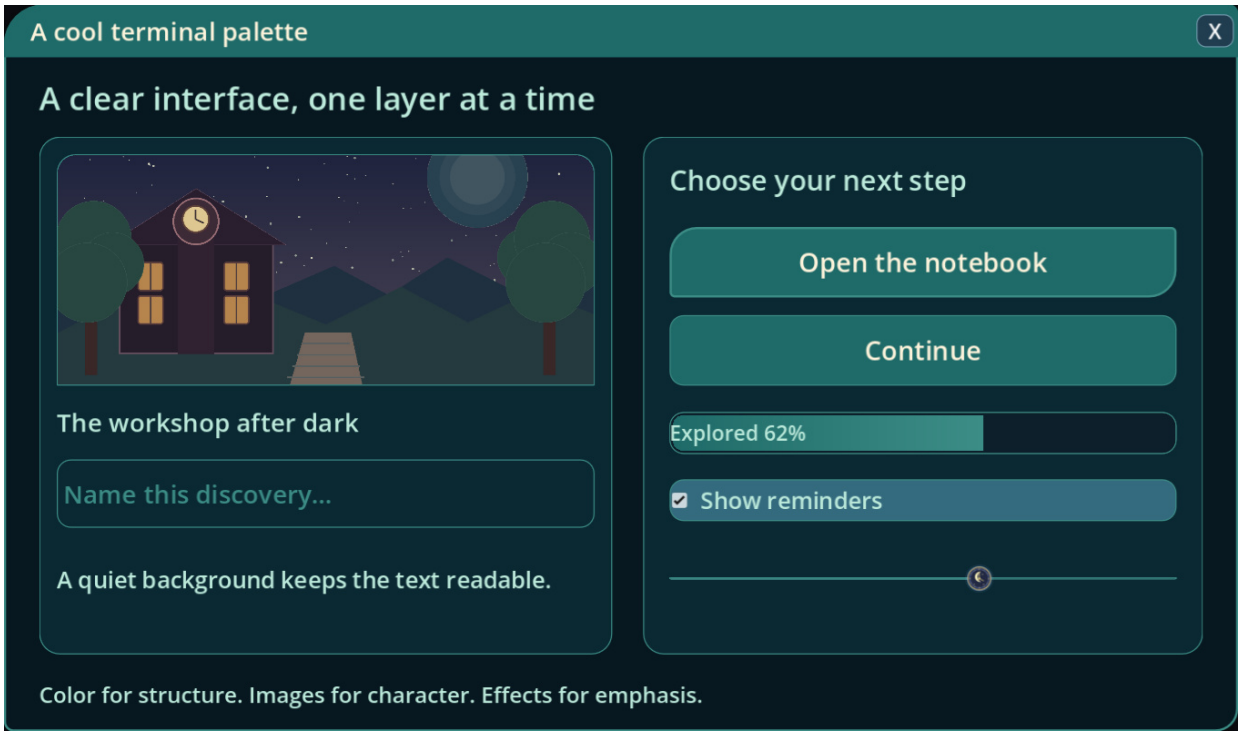

Figure 41. The same controls use a cooler, more technical palette.

Part	Terminal recipe
Window / Panel	#071820 / #0B2933
Text	#BDEBDC
Border / accent	#3D8D87 / #1F6B69
Optional decorative effect	Hologram or mild CRT on an illustration.
Reading layer	Keep instructions, entered text and confirmation captions plain.

Use straight or softly rounded corners consistently. A restrained terminal can feel technical without covering every Label in scanlines. Let artwork and one or two decorative displays carry the stronger effects.

If you provide a reduced-effects option, remove scanline/flicker animation and use the same palette with clean images. The menu’s layout and commands should remain recognizable.

> A terminal style should not make a readable game look broken. Test how it feels after several minutes, not just in one striking screenshot.

Finish the style across the whole game

Review	Questions to answer
Hierarchy	Can the player tell the window title, ordinary choice and main choice apart?
Color	Do the same roles use the same colors across menus?
Corners	Are curves deliberate across window, panels, controls and widget parts?
Typography	Are size, alignment and emphasis consistent?
Images	Do hover/pressed states keep the same object size and framing?
Motion	Do shaders and transitions have a clear purpose and a calmer alternative?
Player context	Can the menu be read over every room where it can appear?

Build a small style record for yourself: palette values, normal/hover/pressed states, a few radii, font sizes, window padding and a short list of permitted effects. Reusing those choices keeps later menus from drifting.

Check the less glamorous screens too: empty inventory, missing save screenshot, long translations, a disabled action and an error or confirmation message. They are part of the interface’s appearance.

> Consistency is not forcing every part to be identical. A save thumbnail, an inventory cell and the outer window can have different curves while still belonging to the same visual family.

18. Troubleshoot color and shape problems

Symptom	Check
A background color does nothing	Opaque background image, Media Asset fallback, or the wrong selected surface.
A border is invisible	Width, color alpha, Border Opacity, and whether the window’s border switch is enabled.
A corner ignores the common radius	That corner has a positive or zero override instead of -1.
Several widget corners change together unexpectedly	Identify root, grid, slot and screenshot records; inspect old normalized values.
Text vanishes behind an opaque Panel	Set explicit drawing layers. Do not rely on equal Z Index for overlapping surfaces.
A transparent control still intercepts input	Its type, enabled state and events; alpha is not an input switch.
A curve crowds the caption	Increase content space/margins or reduce the radius.

Return one surface to a simple opaque color with a visible border. Once you know which rectangle you are editing, restore its artwork, transparency and finish one at a time.

> Before changing five colors, read the inspector heading. The most common wrong-color problem is editing the right field on the wrong element.

Troubleshoot images and states

Symptom	Check
A frame image stretches badly	Fit mode and target proportions. The ordinary image path is not a nine-slice editor.
A picture covers the decorative background	The foreground is opaque; use margins, contain or suitable alpha.
All inventory slots show one object	Object art was assigned as repeated slot background instead of the item icon.
Every save uses the same thumbnail	Save Screenshot Background Image is overriding real per-slot pictures.
Hover makes the icon jump	State-image canvas dimensions, transparent padding, fit and alignment.
Clicked art seems permanent	Distinguish held image state, selected/checked style and a script that explicitly changed the normal image.
A slider handle is hard to use	Painted size, actual handle size, surrounding spacing and touch input.

Use deliberately different normal, hover and pressed test images while diagnosing a state problem. Once the sequence works, replace them with your subtle finished artwork.

> For a release, test press/release both inside and outside the control and after focus changes. A still designer screenshot cannot prove that feedback resets correctly.

Troubleshoot shaders and motion

Symptom	Check
No visible shader change	Correct element, Apply button, active layer, intensity and source detail. A flat surface gives pixel/edge effects little to process.
Text becomes dirty or unreadable	The stack reaches the control’s internal caption; separate art from text.
Tint does nothing	Tint alpha, active effect/finish and whether Heat Haze is using its separate path.
All-None still looks shiny	Metallic Look and Reflection Strength.
Heat Haze loses the original picture	It samples the background; use it alone on a decorative surface.
“Glass” is not blurring the room	Glass Pixel and Frostbite are image treatments, not a general live backdrop-blur widget.
Animation continues with Speed 0	Some effects use time differently. Remove the animated layer for a still alternative.
Room rendering changes do not fix GUI Bloom	The element uses a canvas effect. Test its own settings rather than only scene post-processing.

Use the exact target build and renderer for testing. The examples were checked with the available desktop editor; an exported application on another GPU still needs a visual check.

Performance and runtime rebuilds

A static color and a modest border are cheap visual choices. Animated sampling, outlines and Bloom add work. A complex widget can contain many rendered controls, so its element shader may touch far more than one rectangle.

1. Start with one effect on one decorative image.
2. Test the game at its intended resolution and on the slower hardware you support.
3. Compare the same scene with the effect removed.
4. Reduce effect coverage, layer count or animated decoration if input or frame pacing suffers.
5. Keep a readable low-effects style with the same layout and commands.

In 0.86.3.4, runtime property changes that rebuild inventory/save-style widgets can expose stale inventory-widget references in later refresh/navigation paths. Prefer styling those composite widgets before gameplay, or switching between separately authored views, until your exact build’s dynamic path is verified.

Ordinary Button/Image style examples in this book were exercised separately from rebuilding a live inventory widget. Do not update large sets of widget properties every frame just to animate a hover.

> A screenshot demonstrates appearance, not a performance guarantee. Check a real play session with the busiest menu open, including scrolling, animated icons, model/video content and input.

19. Exact color and corner keys

Job	Authored element key
Normal / hover / pressed backgrounds	background / hoverBackground / pressedBackground
Main text	color
Surface / border / text alpha	opacity / borderOpacity / textOpacity
Border outline	borderColor / borderWidth
Common element curve	radius
Four corners	radiusTopLeft / radiusTopRight / radiusBottomRight / radiusBottomLeft
Metallic finish	metallicStrength
Reflective finish	reflectionStrength / reflectionColor
Main font size / alignment	fontSize / textAlignment
Common GUI window curve	windowFrame.windowBorderRadius
GUI body background	windowFrame.windowBackground / windowBackgroundEnabled

For GUI Window keys, the prefix identifies the stored windowFrame object. In the final row, windowBackgroundEnabled is also inside windowFrame. The script examples show the explicit prefixed form.

The inspector’s appearanceRadius control is an editor alias: it writes radius for an element and windowBorderRadius for the GUI Window. Use the actual authored keys in scripts.

> A property name’s spelling and case matter. Do not infer that changing an editor control name through a runtime action will update the underlying authored field.

Exact image and widget-part keys

Region	Normal image	Fit
Ordinary element	backgroundImageAssetId	backgroundFit
Grid	gridBackgroundAssetId	gridBackgroundFit
Slot	slotBackgroundImageAssetId	slotBackgroundFit
Save screenshot	saveScreenshotBackgroundImageAssetId	saveScreenshotFit

The ordinary element’s hover and press keys are hoverImageAssetId and pressedImageAssetId. Grid uses gridHoverImageAssetId and gridPressedImageAssetId. Slot uses slotHoverImageAssetId and slotPressedImageAssetId. Screenshot uses saveScreenshotHoverImageAssetId and saveScreenshotPressedImageAssetId.

Part	Common curve and corner family
Grid	gridRadius, followed by gridRadiusTopLeft and the other three corners.
Slot	slotRadius, followed by slotRadiusTopLeft and the other three corners.
Save screenshot	saveScreenshotRadius, followed by saveScreenshotRadiusTopLeft and the other three corners.

Slider handle images are sliderThumbAssetId and sliderThumbHoverAssetId. Supported media content uses contentFit and contentMarginLeft/Top/Right/Bottom. Choose these through the typed inspector where possible.

> Media Asset is commonly stored as assetId. Its meaning depends on the element type: a picture, video or model is not the same kind of content as a Button backdrop.

Exact shader and transition structure

A supported shader object is stored on the element as shader. Its layers array contains objects with preset and intensity:

```
{
  "layers": [
    {"preset": "warm", "intensity": 0.45},
    {"preset": "film-grain", "intensity": 0.2}
  ],
  "pixelSize": 4,
  "distortionAmount": 0.01,
  "distortionSpeed": 0.7,
  "colorLevels": 5,
  "edgeStrength": 0.65,
  "tintColor": "#ffffff00"
}
```

This object is the value of shader, not an entire GUI element. It has a maximum of eight recognized layers. An explicit empty shader object removes the element’s ordinary stack and opts out of a legacy fallback.

A GUI stores openTransition and closeTransition separately. Each can include type, duration, color, direction, origin, intensity, randomPool and avoidImmediateRepeat. Use the Transitions dialog for normal authoring rather than pasting unsupported room effect names into these fields.

Layer order is the order in the list. Shared shader parameters are siblings of layers, not extra fields inside each layer.

20. A final practical check

Check	Finished when…
Selection	Every screenshot or edit refers to the intended window/element/part.
Readability	Text stays clear over the lightest, darkest and busiest backgrounds.
Color roles	Primary, ordinary, selected, pressed and disabled states are understandable.
Shape	Corners and margins agree; no text is trapped in a curve.
Artwork	Fit modes and state-image framing are deliberate.
Effects	Each layer has a reason and a clear reset path.
Motion	Menus settle in the right place and have a calmer alternative.
Input	Mouse, touch and keyboard/controller flows are checked where supported.
Release	The exported game is tested on representative target hardware.

Keep a plain copy of the menu while experimenting. It gives you a trustworthy comparison when a shader, image or state change makes something harder to read.

A strong adventure interface helps the player notice a clue, choose an action and return to the story. The style should support those moments without making the player work to read the controls.

> Reference edition: PCA 3D Adventure Studio 0.86.3.4. All figures use actual editor or game-preview captures. No practice files are required.