PCA 3D ADVENTURE STUDIO

GUI Designer

Build the interface your adventure needs, one piece at a time

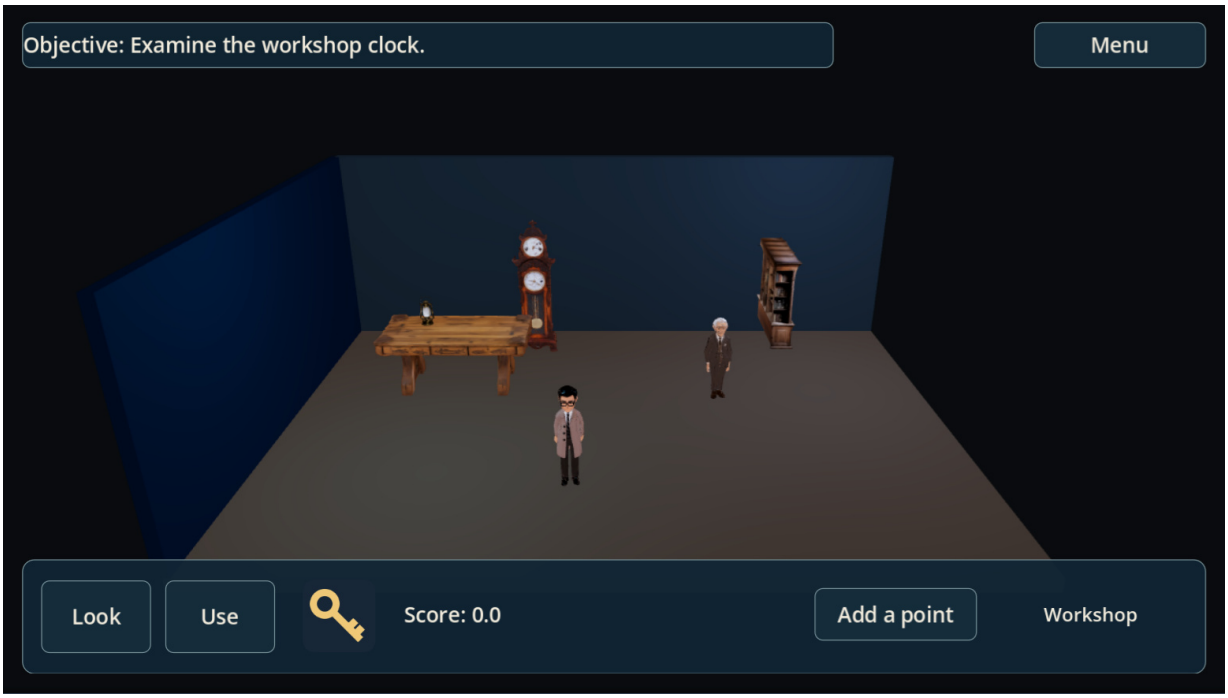

Figure 1. A working adventure HUD over the workshop scene. The menu, verbs, score and objective are authored GUI elements.

Make a clear HUD, open a separate menu, style the controls and connect them to the game. Learn the designer in detail, with short introductions to the specialist widgets.

Version 0.86.3.4 | Windows, macOS and Linux

A practical guide with screenshots from the editor.
No practice project or extra files required.

How to use this guide

Open a project you already know, or use one of the bundled demos. The pictured workshop uses the Clockmaker artwork. It is an example for the instructions, not a project you need to download.

The main walkthrough builds two interfaces: an Adventure HUD that stays on screen, and a Workshop Menu that opens when the player asks for it. Later pages use small examples to explain images, event actions and other controls.

You want to work on...	Read
The window, canvas and basic layout	Chapters 1-7
Text, pictures, colors and corners	Chapters 8-10
Hover, held-click and preview behavior	Chapter 11
Buttons, action lists and when a GUI opens	Chapters 12-15
Tabbed panels and advanced record editing	Chapter 16
Visual nodes and code	Chapter 17
Other controls and the widget overview	Chapters 18-20
HUD/dialogue assignments and final testing	Chapters 21-24

The inventory, save-game, journal, hint, list and carousel elements get a short introduction here. Their detailed content, navigation and internal styling belong in the separate widget tutorials. The basic media and value controls are introduced without turning this into a manual for each one.

> Element properties update live. The GUI record form, Raw JSON editors and some dialogs have their own Apply buttons. Save Project is still what writes your accepted authoring changes to the project.

Use the IDs shown in the examples, or substitute your own consistently. If an example ID already exists, inspect that record before creating another. The screenshots follow 0.86.3.4; the desktop workflow is shared across Windows, macOS and Linux.

Contents

Click a chapter or page number. PDF bookmarks also include the smaller section headings.

How to use this guide . . . 2
1. Think in windows and elements . . . 4
2. Find your way around the designer . . . 5
3. Create a GUI and give it a clear ID . . . 7
4. Size the actual GUI canvas . . . 9
5. Understand X, Y, size and anchors . . . 11
6. Add, select, move and resize elements . . . 13
7. Group and align a tidy layout . . . 15
8. Make labels readable . . . 17
9. Give an element its images . . . 19
10. Style the surface and lettering separately . . . 21
11. Check hover, hold and release . . . 23
12. Make the menu button open a GUI . . . 25
13. Run an ordered list from a button . . . 27
14. Decide when a GUI becomes visible . . . 29
15. Set opening and closing effects . . . 31
16. Make a panel with linked GUI tabs . . . 33
17. Change a GUI from a script . . . 35
18. A first look at text entry . . . 38
19. Add a video or a 3D model . . . 40
20. Recognise the data-driven widgets . . . 41
21. Assign GUIs to the game's system roles . . . 43
22. Test the complete HUD-to-menu flow . . . 45
23. Find the cause when something is missing . . . 47
24. Finish with a clean working interface . . . 49

1. Think in windows and elements

GUI means graphical user interface: the parts of your game the player reads, clicks or types into. The HUD, a pause menu, dialogue choices and a clue notebook can each be a separate GUI.

Each GUI has a window or canvas, then elements inside it. The window establishes the available space. A label supplies text; a button supplies a clickable control; a panel can supply a background; an image displays an imported picture.

Part	Example in this guide	Its job
GUI record	Adventure HUD(adventure_hud)	The whole gameplay interface.
GUI window	1280 by 720, transparent	The HUD canvas and its runtime position.
Element	Menu Button(menu_button)	One control inside that GUI.
Another GUI	Workshop Menu(workshop_menu)	A separate window that can be shown and hidden.
Element event	Menu button: Click	The instruction that opens the menu.

Separate interfaces by purpose

Keep a small gameplay HUD separate from a large menu. You can then hide the menu without hiding the verbs and score. A dialogue interface can have its own fonts and placement without crowding the HUD with every possible control.

A GUI does not need a visible frame. A full-screen transparent canvas is a useful home for a HUD. A smaller canvas with a colored background, title bar and close button suits a movable window.

> The editor theme and UI Size settings change the studio you work in. GUI Designer changes the interface inside your game. Use the game preview to judge its final readability.

2. Find your way around the designer

Open GUI Designer from the designer tabs. Hover over the icons to see their names, and scroll the strip if needed. Select an existing GUI in the selector, or open one from the Database panel.

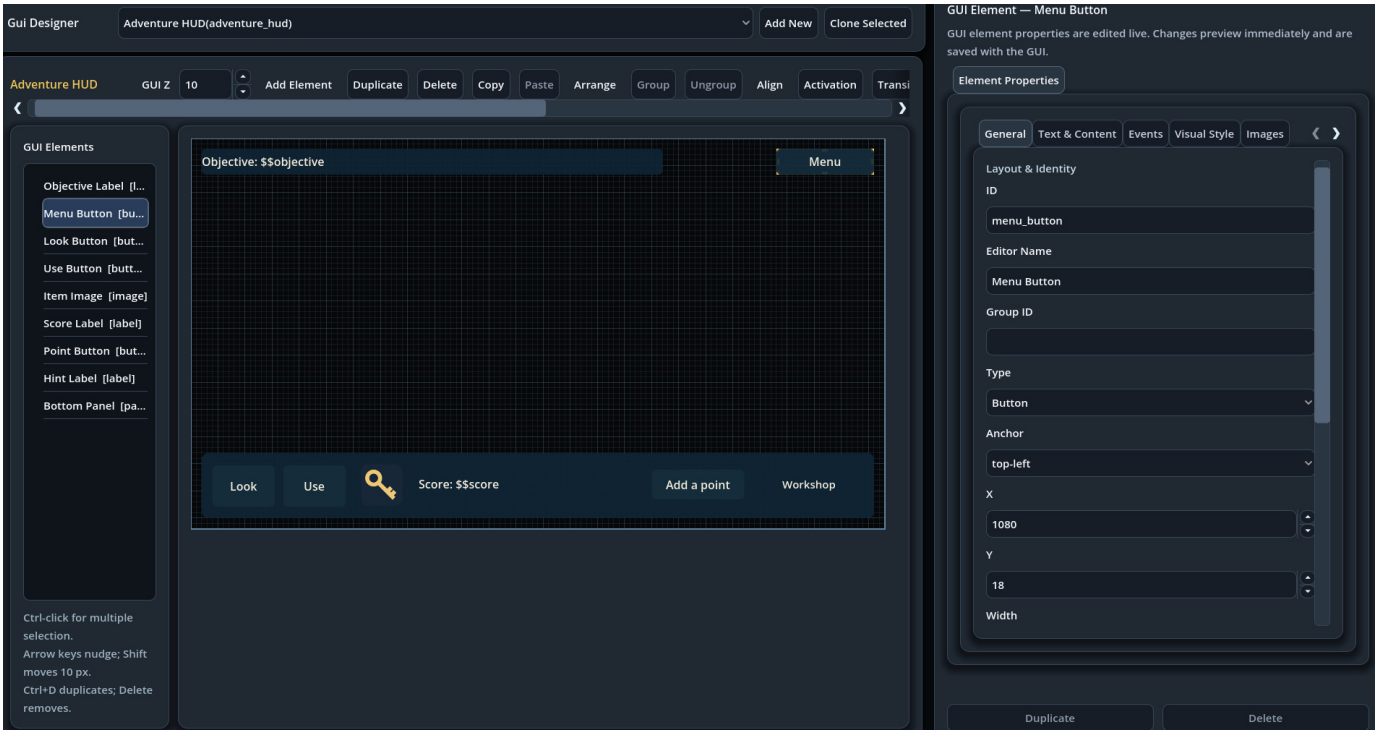

Figure 2. The element list is on the left, the canvas is in the middle and the selected element appears in the Inspector.

Area	What you use it for
GUI selector	Choose a whole interface. Names and IDs are shown together.
Add New / Clone Selected	Create or copy an entire GUI.
Designer toolbar	Add elements, arrange them, edit the GUI window and set preview options.
GUI Elements list	Find a control even when another element covers it on the canvas.
Canvas	Select, move and resize the authored controls.
Inspector	Edit the selected element, or the GUI record when no element is selected.

Read the toolbar in two parts

On a narrow workspace, the toolbar continues sideways. Use its horizontal scrollbar or arrow controls to reach the remaining buttons. A missing button may simply be farther along that row.

Control	What it does
GUI Z	Sets the stacking order of this entire GUI relative to other GUIs.
Add Element	Adds a label, button, image or another supported element type.
Duplicate / Delete	Copy or remove selected elements. These are different from cloning a whole GUI.
Copy / Paste	Copy a selection and paste new, independent elements.
Arrange	Move elements forward or backward in the stack.
Group / Ungroup	Keep selected elements together for movement and resizing.
Align	Line up controls, match their sizes or distribute their centers.
Activation / Transitions	Choose when the GUI opens and how it appears or disappears.
GUI Window	Select the actual canvas and its window settings.
Element Shader	Apply an effect stack to the selected element or elements.
Grid / Snap / grid size	Show guides and control the placement increments.
Zoom / Preview	Change the editor view scale or try the controls without running gameplay events.

> Grid is a visible guide. Snap controls placement. You can hide the grid and still snap, or keep the grid visible while placing elements freely.

The zoom choices are 50%, 75%, 100%, 125%, 150% and 200%. Zoom changes your view of the design; it does not change the saved Width, Height or font sizes.

3. Create a GUI and give it a clear ID

1. Open GUI Designer and click **Add New** beside the GUI selector.
2. With no child element selected, open the GUI record's **General** tab in the Inspector.
3. Set `id` to `adventure_hud` and `name` to Adventure HUD. Use another unique ID if that one already exists.
4. Choose a **Database Folder**, such as Interfaces, if you want to organise the record.
5. Click the record form's **Apply Changes**, then **Save Project**.

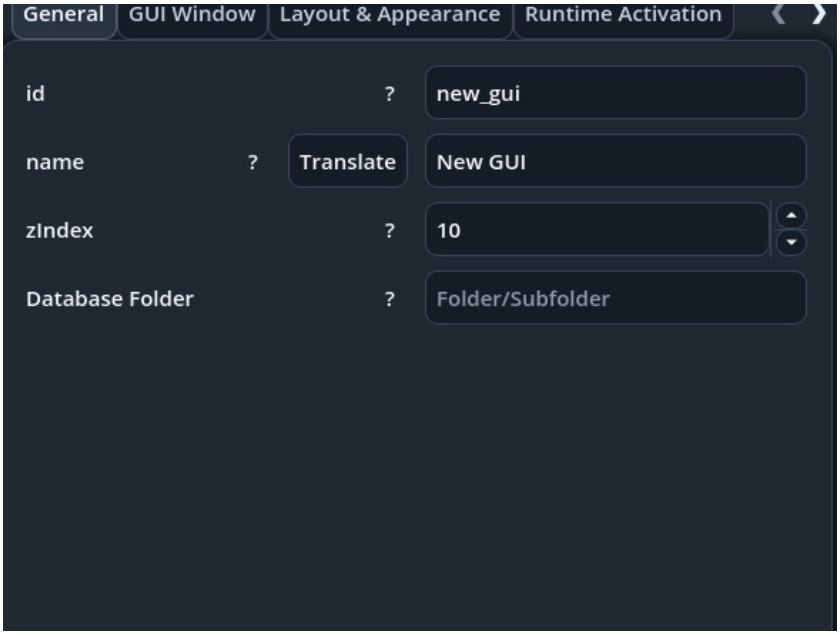

Figure 3. A newly created GUI has its own record form. Its ID and name belong to the whole interface.

Know which kind of ID you are editing

adventure_hud identifies the GUI. menu_button will identify an element inside it. When an action asks for GUI and Element / Group, it needs both levels of that address.

Use short, stable IDs and readable names. For example, Workshop Menu(workshop_menu) is easier to recognise in a selector than a collection of numbered records. Do not assume that changing an ID repairs every hand-written script or event target.

> A new GUI starts hidden and waits to be called. You will choose Visible at Start for the HUD and keep the menu trigger-controlled later in the walkthrough.

Saving, copying and deleting the right thing

What you selected	What the command affects
A GUI in the top selector	Add New and Clone Selected deal with whole GUI records.
An element in the canvas or list	The designer toolbar's Duplicate, Copy, Paste and Delete deal with elements.
GUI Window	The real canvas and its window settings, rather than an extra panel element.

Live fields and explicit Apply buttons

When the Inspector says GUI Element, its normal property fields update the control live. You do not need a general Apply button for a font size, image choice or corner radius. A short series of edits is grouped into an undo step.

The whole-GUI record form is different. It has Apply Changes. The Raw JSON tabs also have Validate and Apply buttons, and dialogs such as Activation, Transitions and Element Shader have their own confirmation buttons. Accept those edits, then save the project.

Copy a useful starting point

1. Select a GUI that already has a useful layout.
2. Click **Clone Selected**, then inspect the new GUI ID and name.
3. Change the copy's purpose, activation and any event targets that should refer to the copy.
4. Save and test it separately from the original.

Cloning does not mean every button now points to the new GUI. A copied Close button may still name the old one. Review those targets deliberately.

To remove an entire GUI, select its record and use **Delete** at the bottom of the Inspector. The Delete button in the designer toolbar removes the selected child elements instead.

> Before deleting a whole GUI, check system assignments, buttons, scripts and tab links that name it. Undo can recover a mistaken edit while it remains in the editor history, but it is better to remove the references as part of the same change.

4. Size the actual GUI canvas

Click **GUI Window** in the designer toolbar. The canvas boundary becomes the selected item, and the Inspector opens the GUI Window tab. This is the space your elements belong to; adding a large Panel is not the same as resizing it.

1. For the HUD example, set **GUI Width** to `1280` and **GUI Height** to `720`.
2. Set **Runtime X** and **Runtime Y** to `0`.
3. Leave **Use Window Background Color**, **Show Border Frame** and **Show Title Bar** off for a transparent HUD.
4. Apply the GUI form and save.

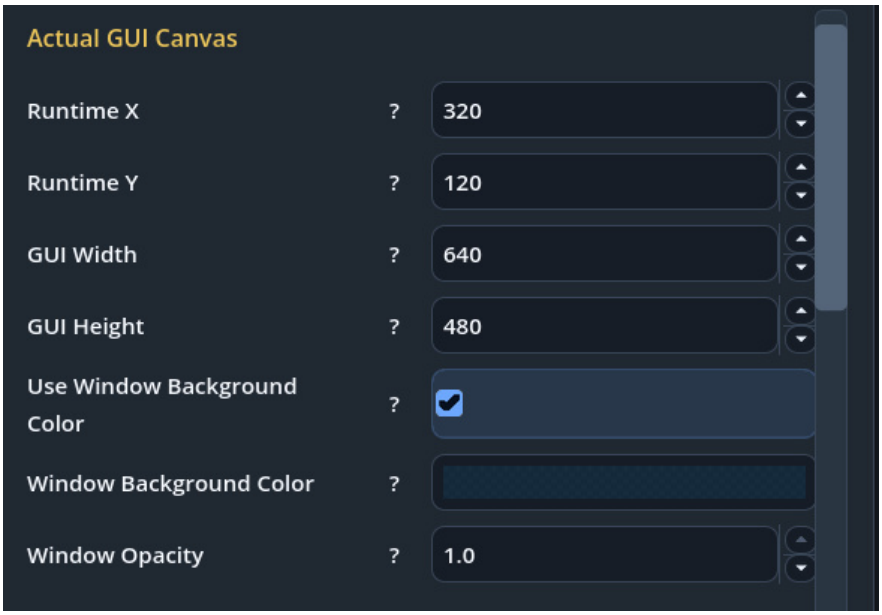

Figure 4. The same controls can make a smaller menu. Here the menu canvas is 640 by 480, with a visible background and border.

Setting	Meaning
Runtime X / Y	The authored position in the game viewport. An entire-GUI anchor can override this placement.
GUI Width / Height	The space available inside this GUI. Elements use coordinates within it.
Window Opacity	The window surface opacity. Check text readability separately.
Border frame controls	Enable the outline, choose its color and width, then set its overall radius.

The designer shows the GUI at the editing canvas origin. Runtime X and Y describe its game position; they are not a way to pan the editor view. You can also resize the selected GUI boundary with its handles.

Give a menu a title bar and close button

Create a second GUI, ID workshop_menu, Name Workshop Menu. A 640 by 480 window is a useful starting size for this example. Enable its background and border, then add the title controls you want.

1. Select **GUI Window** and enable **Show Title Bar**.
2. Set **Title Text** to Workshop Menu. Choose the title height, background, font and text alignment.
3. Enable **Drag GUI From Title Bar at Runtime** if the player should move the window.
4. Enable **Show Close Button**, choose text or image content, and set its location and size.
5. Apply and save, then check the title and close button in a gameplay preview.

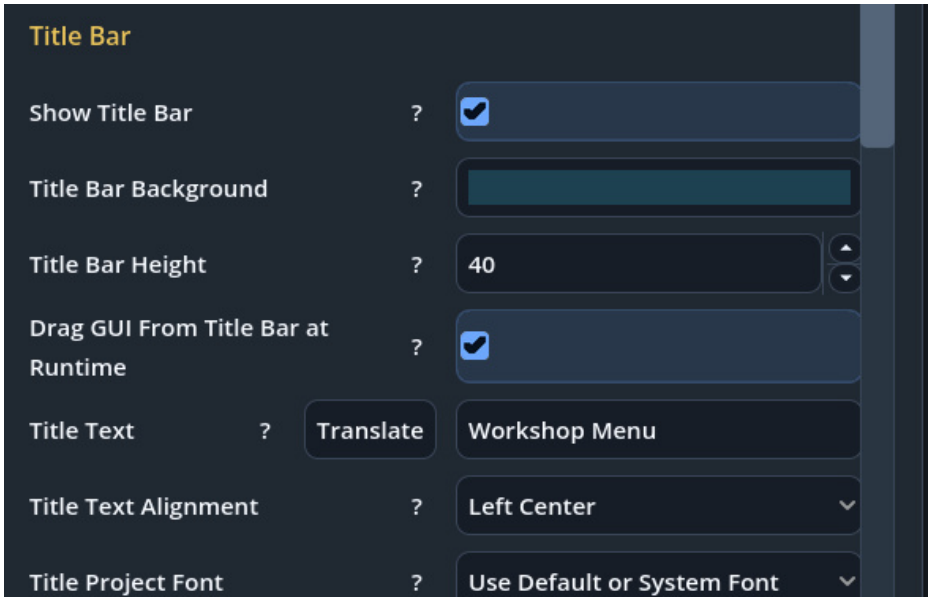

Figure 5. Title-bar settings belong to the GUI window. Keep the first content element below the title strip.

Part	Useful controls
Title	Height, background, text, font size/weight, decorations, color, opacity and alignment.
Title icon	Show Title Icon, imported image asset, location and size.
Close button	Text or image, location, width/height, text styling and normal/hover/pressed colors.

Prefer imported project images for title and close icons when distributing a game. A file path that only exists on your computer is harder to keep portable. The close control hides this authored GUI; it is not the same as quitting the game.

5. Understand X, Y, size and anchors

An element's X and Y are offsets from its chosen Anchor inside its own GUI. Width and Height are its saved size in design pixels. They do not change when you zoom the editor view.

Anchor example	Settings for a 1280 by 720 GUI	Result
top-left	Width 180, Height 48; X 20, Y 18	The element begins 20 pixels from the left and 18 from the top.
top-right	Width 180, Height 48; X -20, Y 18	Its right edge stays 20 pixels inside the GUI's right edge.
center	Width 180, Height 48; X 0, Y 0	The element is centered in the GUI.
bottom-right	Width 180, Height 48; X -20, Y -20	It sits 20 pixels inside the bottom and right edges.

1. Select Menu Button in the HUD and open **General**.
2. Choose **top-right** for Anchor.
3. Set X to `-20`, Y to `18`, Width to `180` and Height to `48`.
4. Check the result on the canvas. Negative offsets here move the element inward from the right or bottom.

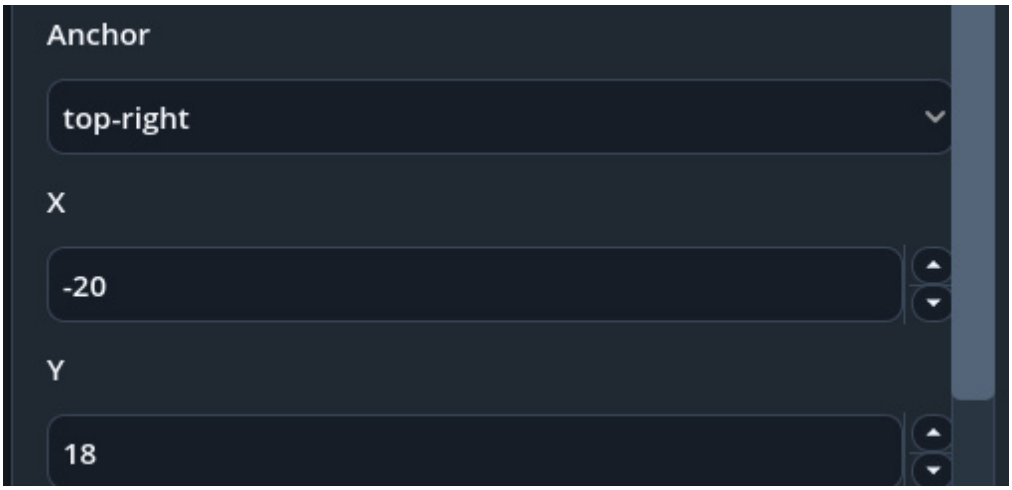

Figure 6. The anchored menu button uses an inward X offset. A large positive X with top-right would push it outside the GUI.

Choose the anchor, then enter the offsets for that anchor. These are attachment points, not an automatic stretching layout. Test long text and different window sizes rather than assuming an anchored label will grow to fit.

Anchor the whole GUI separately

An element anchor works inside one GUI. **Anchor Entire GUI** positions the complete interface inside the game viewport. Use the latter to center a menu without recalculating every child element.

1. Select the Workshop Menu, clear the element selection or click GUI Window, and open **Layout & Appearance**.
2. Choose **Center** for **Anchor Entire GUI**.
3. Set **Anchor Offset X** and **Anchor Offset Y** to `0`.
4. Apply and save. Run the game and open the menu to check its placement.

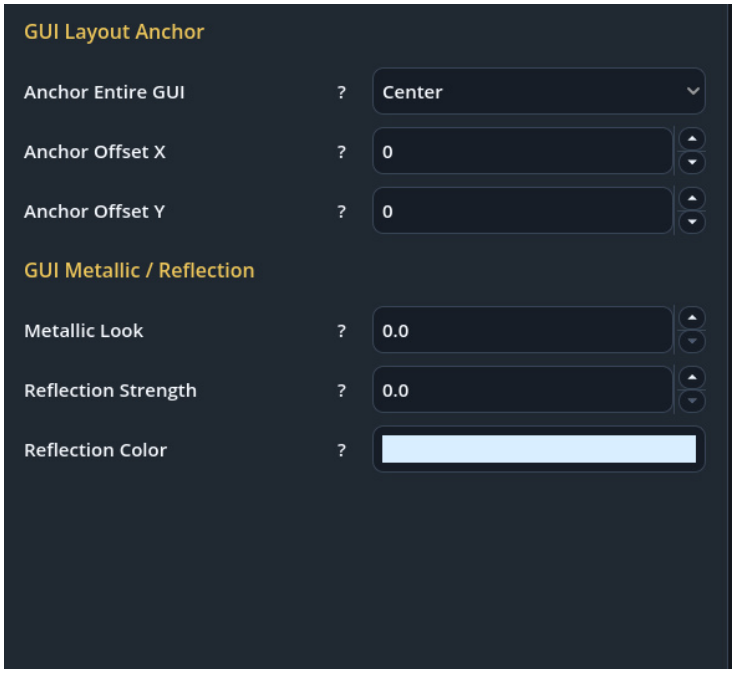

Figure 7. Center places the whole menu in the viewport. Use Authored Coordinates when you want Runtime X and Y to determine its position.

Design size, game size and editor zoom

The walkthrough uses a 1280 by 720 game design. The HUD uses that whole area; the menu uses a smaller canvas. The runtime presents the design within the game viewport, while editor zoom only changes how much fits on your workspace.

Leave comfortable margins around controls. A menu that fits at one resolution can still contain clipped text, small hit areas or overlapping elements after you change its canvas size or translate the wording.

The surface and reflection controls in this tab concern the GUI's visual finish. They do not make a live mirror of a character or a room. Element appearance is covered separately in Chapter 10.

6. Add, select, move and resize elements

1. Choose the Adventure HUD and click **Add Element**.
2. Choose **Label**. The new control appears on the canvas and in GUI Elements.
3. In **General**, change ID to `objective_label` and Editor Name to Objective Label, using a unique ID within this GUI.
4. Set its X, Y, Width and Height, then use **Text & Content** for the words.
5. Save the project after the live edits have been accepted.

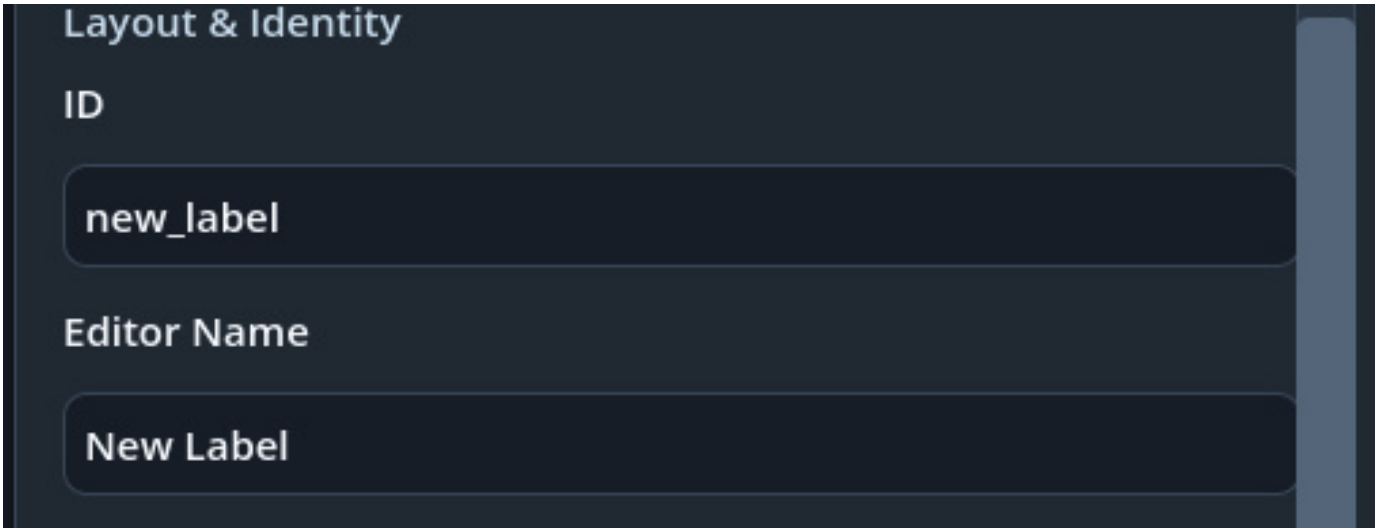

Figure 8. A new element has its own General tab. Editor Name is a label for you; its ID is used by actions and scripts.

Task	How to do it
Select a covered control	Choose it from GUI Elements rather than clicking through other artwork.
Move	Drag the selected control, or enter exact X and Y values.
Resize	Drag an edge or corner, or enter Width and Height. The editing safety minimum is 12 by 12.
Select several	Ctrl-click adds or removes controls from the selection.
Fine movement	Focus the canvas and use arrow keys. See the note below for Shift.

An arrow nudge is one design pixel. Shift uses ten pixels when Snap is off; with Snap on it uses the current grid step. The default grid step is ten, so the usual starting behavior is the same.

Build a small HUD with known measurements

Use the following as a starting arrangement, not a required template. Begin with a transparent 1280 by 720 HUD. These elements use top-left unless another anchor is specified.

Element	Type	X, Y	Width, Height
objective_label	Label	20, 18	850, 48
menu_button	Button	top-right: -20, 18	180, 48
bottom_panel	Panel	20, 580	1240, 120
look_button	Button	40, 602	115, 76
use_button	Button	170, 602	115, 76
item_image	Image	312, 599	80, 80
score_label	Label	420, 615	260, 46
point_button	Button	850, 610	170, 55

Set the panel's Z Index to 0 and the controls above it to 1. Give the buttons the texts Look, Use, Menu and Add a point. Use Text & Content to set the label text; the dynamic values are connected in Chapter 8.

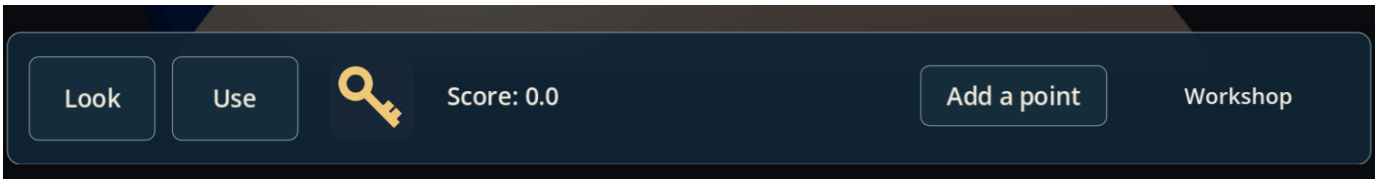

Figure 9. The lower part of the example HUD during play. The panel supplies the backing; the controls remain separate elements.

Why the panel goes behind the controls

A panel is useful for a shared background. Ordinary controls are not automatically children of that panel just because they overlap it. Use their own positions, and group them when you want to move the arrangement together.

> Keep Preview off while building the layout. If a click operates a control instead of selecting or dragging it, switch back to editing mode.

7. Group and align a tidy layout

Use groups for pieces that should travel together, such as a verb icon and its label. Use alignment for pieces that should line up but can still be edited independently.

1. Ctrl-click the controls you want to keep together.
2. Click **Group**. The elements receive a shared Group ID.
3. Move a grouped element to move the group. Resizing a grouped selection transforms the group's positions and sizes proportionally.
4. Use **Ungroup** when you need to edit the members separately, then group them again if useful.

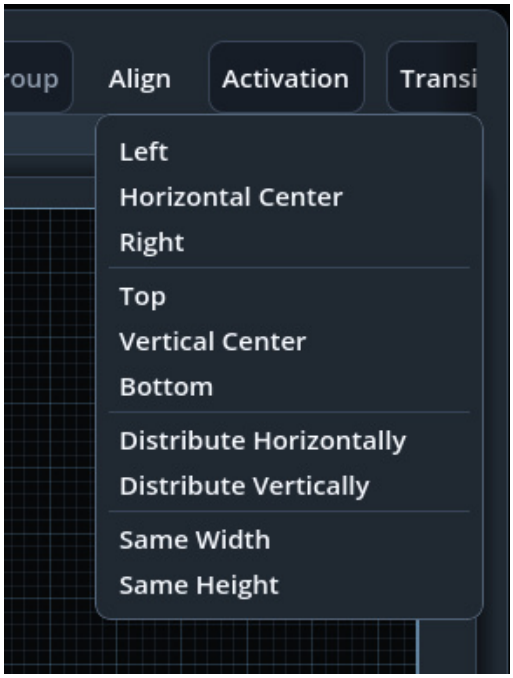

Figure 10. Align offers edge, center, size and distribution commands. The control you make primary matters for multi-selection alignment.

Align command	Reference it uses
One element selected	The GUI canvas. For example, Horizontal Center centers it in the GUI.
Several elements selected	The primary selected element. Select the intended reference last.
Same Width / Same Height	Matches the primary control's dimension.
Distribute Horizontally / Vertically	Needs at least three controls. It spaces their centers between the first and last.

Copy, Paste and Duplicate give copied groups new Group IDs so moving the copy does not move the originals. While several elements are selected, the detailed single-element form is hidden; use Group, Align or Arrange, or return to one selection.

Control what sits in front

There are two stacking levels. **GUI Z** orders whole interfaces. An element's **Z Index** and the **Arrange** commands order controls inside its GUI.

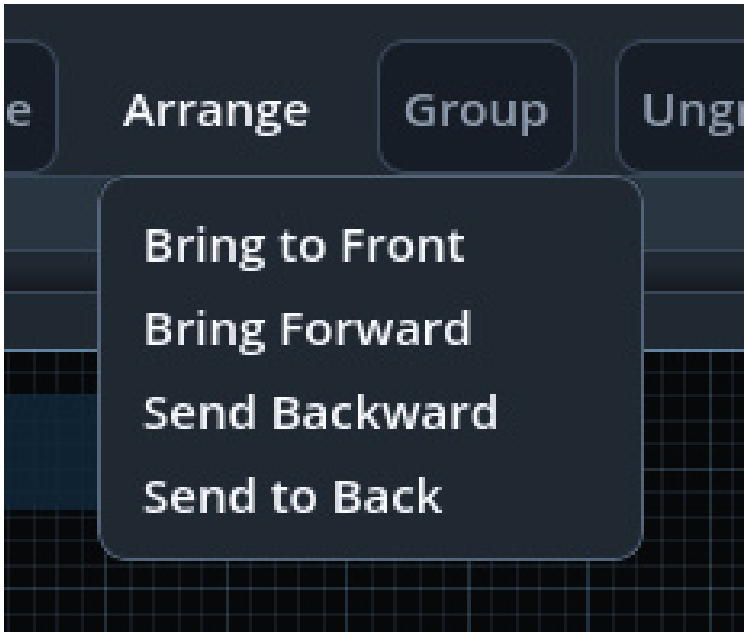

Figure 11. Arrange changes the selected elements' Z Index. A background panel should sit behind its text and buttons.

Command	Effect
Bring to Front	Moves the selection above the highest existing element Z value.
Bring Forward	Adds one to each selected element's Z value.
Send Backward	Subtracts one from each selected element's Z value.
Send to Back	Moves the selection below the lowest existing element Z value.

Use a modest set of values: the background at 0, controls at 1 and a deliberate overlay above those. Raise GUI Z for a separate menu that must appear above the HUD.

Visible and Enabled answer different questions

Visible controls whether an element is shown. Enabled controls whether it can be operated. A disabled button can remain visible so the player knows the option exists; a hidden button is removed from view.

> A high Z value does not put an off-canvas element back inside the GUI. If something is missing, check its anchor, offsets, size and visibility as well as its stacking order.

8. Make labels readable

1. Select Objective Label and open **Text & Content**.
2. Enter the text in **Text**. For now, use a short instruction such as Examine the workshop clock.
3. Choose a Font, Font Size and alignment that fit the control.
4. Set the text color and Text Opacity. Save, then read it in the game preview over the actual room.

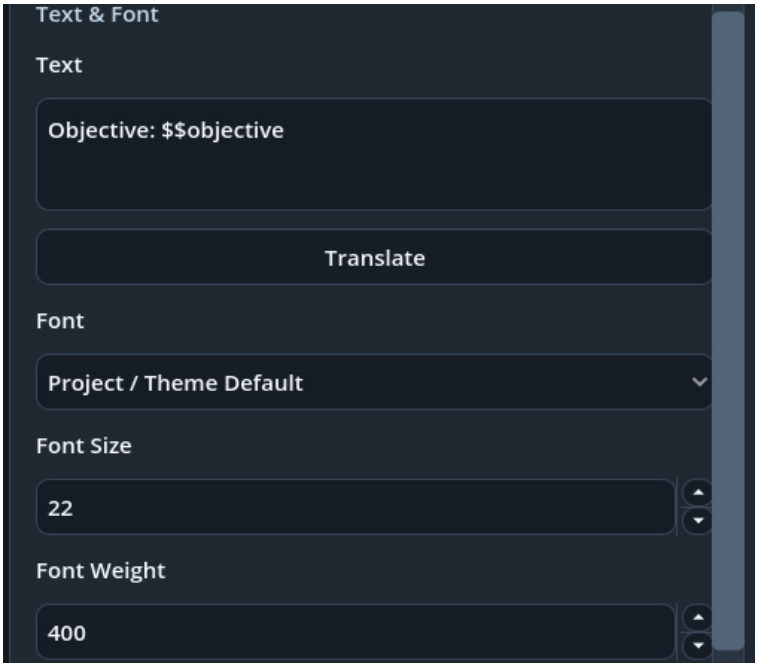

Figure 12. Text and font controls are shared by text-capable elements. A button's caption uses the same basic styling tools.

Setting	Use it for
Font	A project font, a system font or the default font. Imported project fonts give more predictable shipping results.
Font Size / Weight	The size of the lettering and the weight requested from the font.
Bold / Italic	Emphasis, where the font supports it.
Underline / Strikethrough	Text decoration.
Text Color / Text Opacity	The lettering itself, independently from the element surface.
Text Alignment	Left, center, right, left-center or right-center placement.

If text clips, first give it enough Width and Height. Test a longer sentence before shrinking the font. Fonts installed on your computer may not be installed on another player's machine.

Display variables and translate the wording

A label can show live game values. Use two dollar signs followed by the exact variable ID, such as $$score. The designer may show the token while you author; check the resolved result in gameplay.

Element text	Example during play
Score: $$score	Score: 1.0
Objective: $$objective	Objective: Find the missing winding key.
@PlayerName	The active player character's name.

1. In Variable Designer, define `score` as Number starting at 0 and `objective` as String starting with your instruction.
2. Set Score Label's Text to `Score: $$score`.
3. Set Objective Label's Text to `Objective: $$objective`.
4. Run an action that changes a value, then check the label in a gameplay preview.



Figure 13. The score label after the Add a point event. A whole amount may be displayed with .0 after numeric arithmetic.

Translate text, keep IDs stable

Use the **Translate** button beside the Text editor to work with that player-facing wording in the translation workflow. Keep variable tokens and IDs intact. Translating the sentence does not mean translating score into a different runtime key.

Try longer translated labels and accented characters with your chosen font. Buttons need enough space for the translated caption, not just the shortest English wording.

> The element's Editor Name is for finding it in the studio. Text is what the player reads. Changing one does not automatically change the other.

9. Give an element its images

Import your pictures through Asset Designer first. Then select the GUI element and open **Images**. Choose from the project assets rather than typing a path to a file elsewhere on your computer.

Images field	What it supplies
Media Asset	The foreground picture of an Image, the content of a model/video element, or a normal background on an ordinary control.
Background Image	An explicit backdrop. On an ordinary control it takes priority over an image supplied through Media Asset.
Hover Image	The pointer-over artwork. On an Image element it replaces the foreground picture.
Clicked Image	The artwork used while a left press is held. It is a temporary state.
Background Fit	How the backdrop is fitted into the element frame.

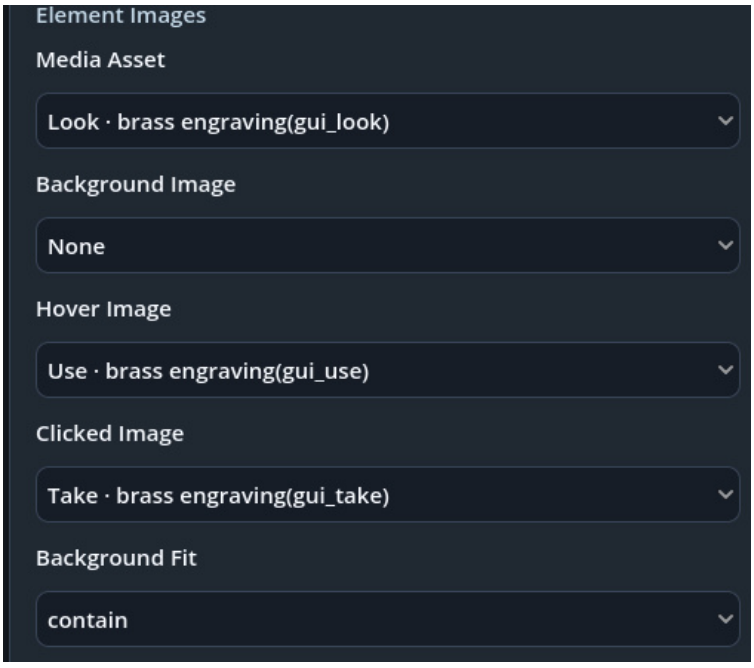

Figure 14. The image-state fields are together in Images. This example uses visibly different Clockmaker icons to make the states easy to test.

For an Image element, Media Asset and Background Image have separate jobs. A key icon can remain the foreground while a paper texture sits behind it. The foreground has its own Content Fit in Type Settings.

> The state pictures in this guide are a test, not a rule for artwork. For a finished game, use matching normal, hover and clicked versions so the control still looks like the same object.

Fit the picture without changing the frame

Select an Image and open **Type Settings**. **Content Fit** controls the foreground picture; the four margins inset that picture inside the same element frame.

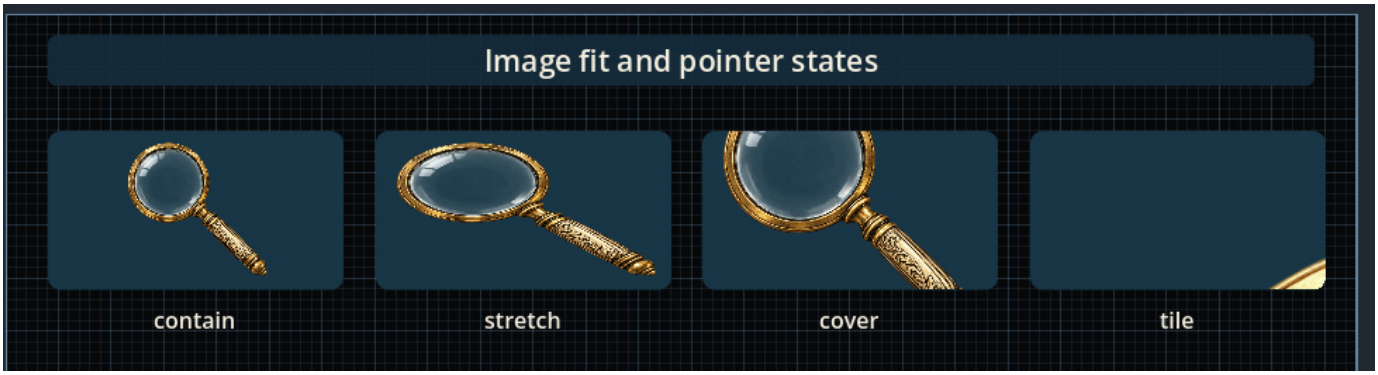

Figure 15. The same imported picture in four frames. Contain, stretch, cover and tile make different choices about empty space, cropping and repetition.

Fit	What happens
contain	Shows the whole picture at its original proportions. Empty space may remain around it.
stretch	Fills the frame exactly. The picture can look stretched if the shapes do not match.
cover	Fills the frame while keeping proportions. Some edges of the picture may be cropped.
tile	Repeats the image across the available area. Useful for a repeatable texture.

Use the right setting for the right layer

Changing Background Fit does not change an Image element's foreground Content Fit. Likewise, increasing a content margin does not move the element itself. Use General for the frame's position and size.

For a decorative animated image, use **Animation Frame Asset IDs** and **Image Animation FPS** in Type Settings. Enter the imported frame IDs in the intended order. Keep this separate from a character portrait's speech and lip-sync setup.

Check alpha transparency in the source artwork as well as the chosen fit. A solid rectangle baked into the picture remains a solid rectangle even if the element behind it has a transparent background.

10. Style the surface and lettering separately

Open **Visual Style** for the selected element. It controls the surface, border, colors and corner shape. Text & Content still controls the lettering.

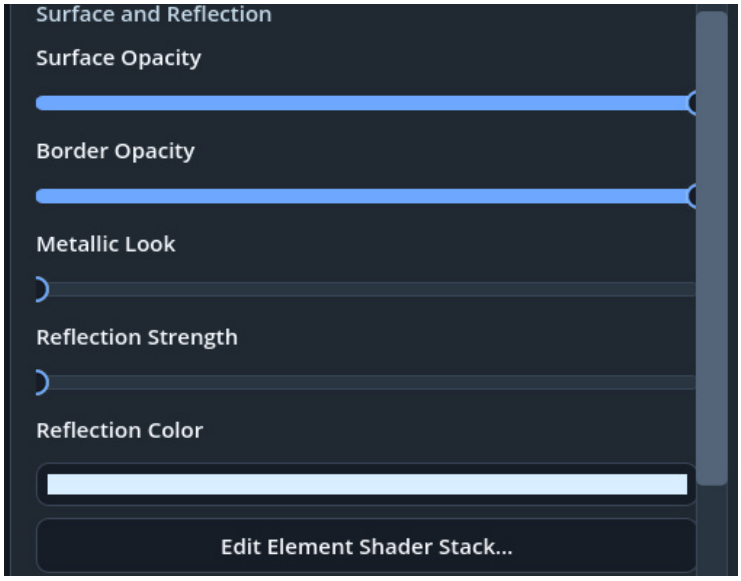

Figure 16. Visual Style starts with surface and reflection controls, then continues into the ordinary shape and color settings.

Control	What to check
Surface Opacity	The surface strength. Do not use it as a substitute for Text Opacity.
Border Opacity / Border Color	How visible the outline is and which color it uses.
Background Color	The control's normal surface color.
Hover Background	The pointer-over surface color.
Pressed Background	The held-press surface color.
Metallic Look / Reflection Strength / Color	A GUI surface finish. It is not a live mirror of the scene.

Use the color picker to choose the hue and alpha where available. A bright room behind a translucent HUD can make text disappear, so judge the combination over the room you actually intend to use.

The shared typed form exposes the border color and opacity. A normal element's borderWidth can also be stored or changed through advanced JSON/property actions; the main GUI window has its own explicit Border Width field.

Give each corner its own radius

Border Radius supplies the overall curve. The four corner fields can inherit that value or override it independently. All amounts are in design pixels.

Corner value	Meaning
-1	Use the overall Border Radius for this corner.
0	Make this corner square.
A positive number	Use that curve size for this corner.

1. Add or select a Panel and open **Visual Style**.
2. Set **Border Radius** to `12`.
3. Set **Top Left Radius** to `50` and **Bottom Right Radius** to `35`.
4. Set **Top Right Radius** and **Bottom Left Radius** to `0`.
5. Check the panel at normal size, then save the project.



Figure 17. The example panel has two rounded corners and two square corners. An override affects only its named corner.

Use this for a speech-panel edge, a notebook card or a deliberately uneven menu shape. If one corner does not respond to the overall value, check whether it still has its own positive override.

> The outer surface and background shape can be rounded. Video/model content and some shader output are not automatically masked into the same rounded shape. Grid widgets also have separate internal surfaces; those are only introduced in Chapter 20.

11. Check hover, hold and release

Assign a normal image, a Hover Image and a Clicked Image that you can tell apart. Enable the designer's **Preview** switch, then move over the control, hold the left button and let go.



Figure 18. Normal: the original artwork is shown before the pointer enters.



Figure 19. Hover: the separate hover artwork appears.

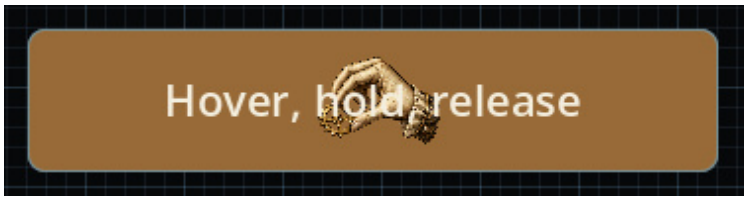

Figure 20. Held: the Clicked Image is visible while the press is held.



Figure 21. Released: the original artwork returns. Move off and back on to enter the hover image state again.

Clicked Image is a pressed-state picture, not a permanent selection picture. The button may still have focus or hover coloring, but the configured held-click artwork should not remain stuck after release.

An omitted state image uses an available fallback rather than requiring three pictures for every element. Hover is a mouse interaction; provide a clear normal and pressed appearance for touch users too.

Use Preview for appearance and Play for events

Test	Designer Preview	Gameplay preview
Hover and held-click artwork	Try the visual states.	Check the same states over the real game.
Buttons' gameplay actions	They are not executed here.	Click to test scripts, variables, show/hide and action lists.
Text entry and simple controls	Try the local control behavior.	Check parser submission, shared values and game integration.
Linked tab GUIs	The preview can show an Embedded GUI hint.	Check the actual linked page content.
Inventory/save data	May use editor sample content.	Uses the running game's items and saves.
Model viewer input	Can be tried when enabled.	Check the intended runtime behavior.

A simple two-stage test

1. Turn Preview on. Check normal, hover, held and released appearance.
2. Turn Preview off before selecting, dragging or editing controls.
3. Save the project and use **Play / Restart**, or Test Scene where appropriate.
4. Click the control in the game. Check the actual result as well as the artwork.

The example state button has a score-changing event. Clicking it in Designer Preview did not change the game's starting score; the fresh gameplay test still began at zero. The gameplay Add a point button did update the live score.

> A button that looks responsive in Designer Preview can still have a wrong event target. A button that does not open a menu there is not proof of a broken event. Use the appropriate test for each job.

12. Make the menu button open a GUI

1. Select Menu Button in Adventure HUD and open **Events**.
2. Find the **Click** section, below Hover.
3. Choose the shared catalog action **Show GUI**.
4. In its **GUI** parameter, select Workshop Menu(workshop_menu).
5. Save and start a gameplay preview. Click Menu and check that the separate window opens.

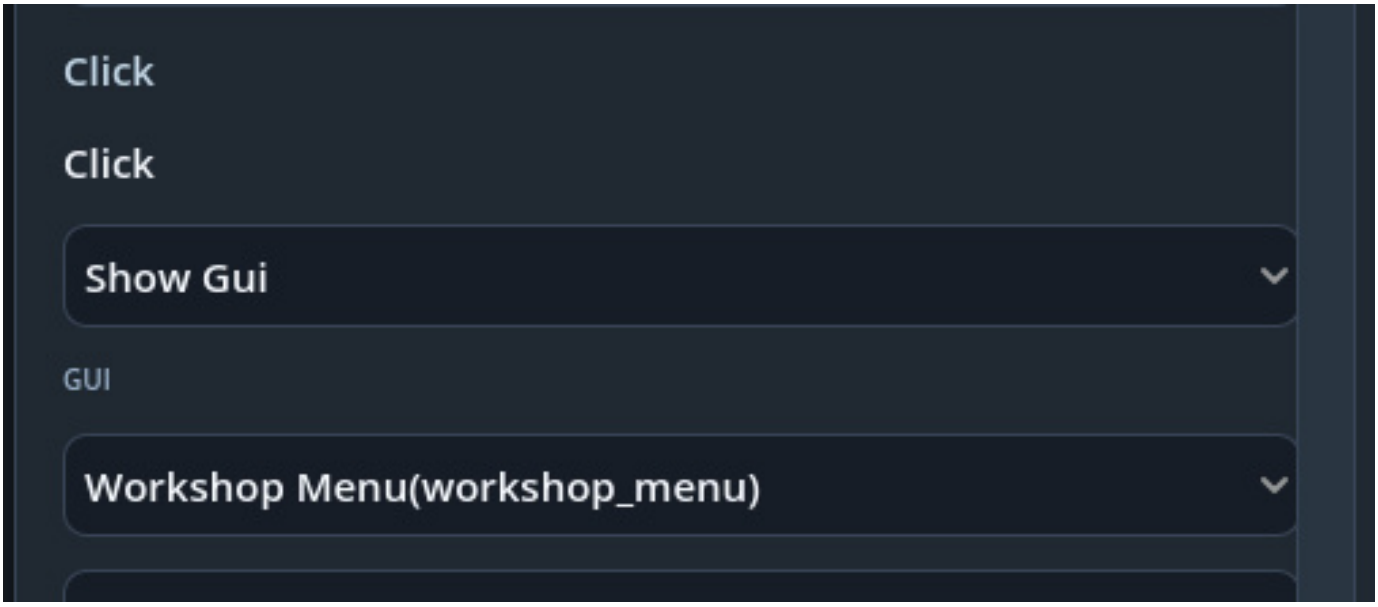

Figure 22. The catalog Show GUI action has a GUI reference parameter. Select the named record rather than relying on its display name as an ID.

Give the menu a return button

Inside Workshop Menu, add a Button with ID return_button and text Return to game. Give its Click event the **Hide GUI** catalog action, targeting workshop_menu. Save and test opening and closing the menu.

A control with only decorative images is not automatically a game command. The Events tab supplies that connection. If you use an Image as a clickable icon, give it an event deliberately.

> For a custom overlay, Show GUI displays that GUI. It does not automatically mean “pause every timer and animation in the game”. Modal/input behavior is a separate concern, covered in Chapter 16.

Understand the five mouse events

Event	Use it for
Hover	A pointer-enter response. Keep it short and avoid rewards that can repeat whenever the pointer re-enters.
Click	The ordinary left-click action.
Double Click	A separate double-left-click response.
Right Click	An alternate command or context action.
Double Right Click	A separate double-right-click response.

Two similar-looking kinds of choice

The list includes built-in behaviors such as show-gui, hide-gui, toggle-gui and set-verb, as well as the shared action catalog such as Show GUI, Set Variable and Change GUI Element Property.

Chosen kind	Where its parameters go
Built-in show-gui	Put the GUI ID, such as workshop_menu, in Click Target.
Catalog Show GUI	Use the GUI parameter in the action editor.
Built-in set-verb	Use the Click Verb selector. Choose Look or Use from the defined verbs.
Catalog action	Edit that action's displayed parameters, just as in an action list.

For the HUD, give look_button a set-verb Click behavior with Look as Click Verb, and do the same for use_button with Use. During play, choosing a verb prepares the next world interaction; it does not automatically use an arbitrary scene object.

> A double-click normally includes an earlier single click. Do not place unrelated irreversible rewards on both events and assume only one will run. On touchscreens, keep important commands available through ordinary taps.

The Legacy Click Action field remains for older/custom action strings. Prefer the explicit Events setup for new controls, and check for old behavior when converting an existing interface.

13. Run an ordered list from a button

Use Additional Event Actions when a single click needs several steps. The primary behavior or catalog action runs first; the additional list runs afterward, in order.

1. Select Add a point and open its Click event.
2. Choose **Action List** as the primary choice, so this example runs only the list.
3. Expand **Additional Event Actions** and click **Add Next Action**.
4. Choose **Change Variable**, select Score(score), choose Operation `add` and enter Value `1`.
5. Save, play and click once. The live score should rise from 0 to 1.

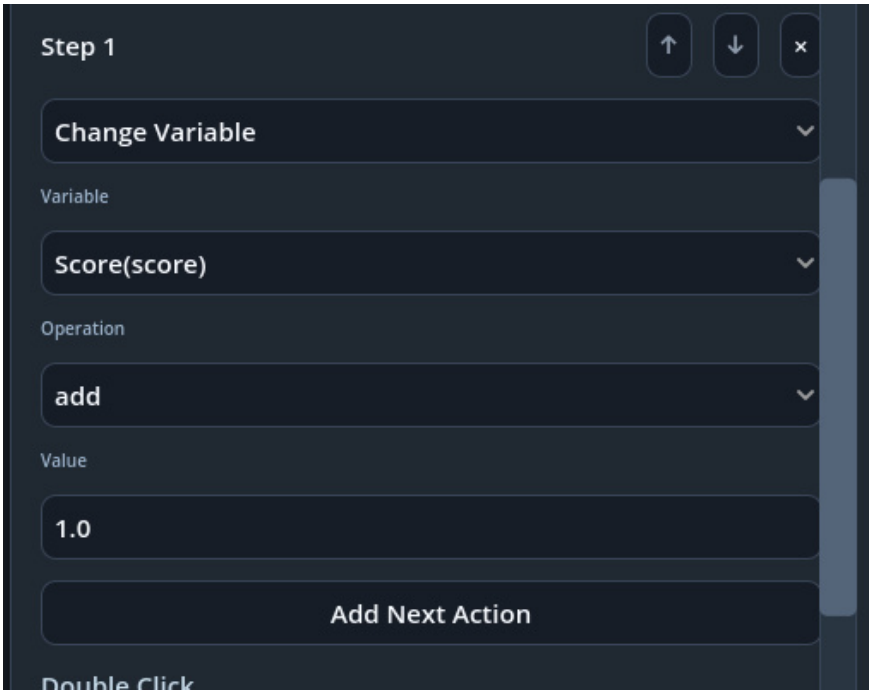

Figure 23. The additional list has its own action selector, parameter fields and step controls.

Control	Meaning
Add Next Action	Append another step.
Up / Down arrows	Move a step in the sequence.
The x control	Remove that step.
None plus a list	Allows a list without a primary behavior, subject to any old legacy Click Action being cleared.

If an action exposes a Wait option, choose it according to the intended order. A long-running action that is not told to wait may let the next step begin while its visual result is still happening.

Update the objective, then close

Add an Update objective button to Workshop Menu. Its Click event will replace the instruction and then hide the menu. This is a complete small example of a GUI changing gameplay state.

1. Choose **Action List** for the button's Click event and open Additional Event Actions.
2. Add **Set Variable**, choose Current Objective(objective), and set its text to `Find the missing winding key.`
3. Add **Hide GUI** after it, targeting Workshop Menu(workshop_menu).
4. Save, play, open the menu and press Update objective.
5. Check that the menu closes and the HUD objective changes.

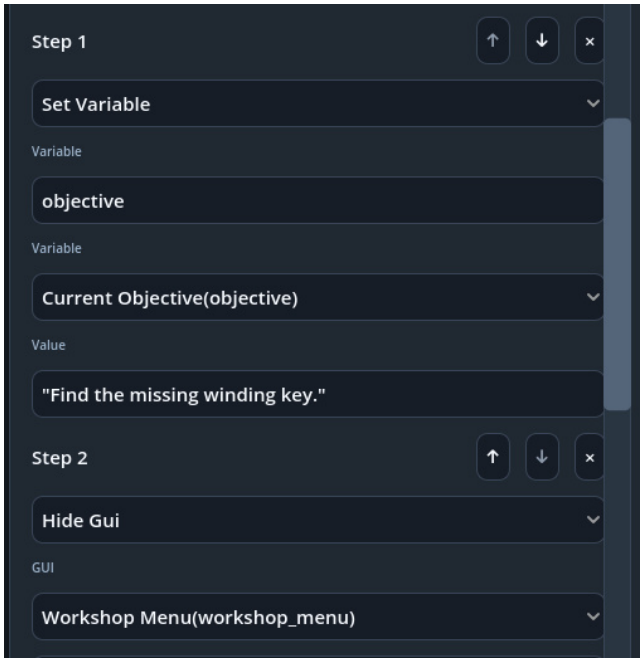

Figure 24. The sequence is stored on the button event. The second action follows the state change.

Order	Action	Result
1	Set Variable: objective	The shared instruction changes.
2	Hide GUI: workshop_menu	The menu disappears using its close transition.

Changing objective and displaying objective are separate jobs. The action writes the variable; the HUD label's $$objective token shows it. This is why the same value can be used by more than one GUI or script.

> Use plain text in a dedicated text field. In a general JSON-style value field, put Strings in quotes and leave numbers and Booleans unquoted.

14. Decide when a GUI becomes visible

Click **Activation** to open GUI Runtime Activation. A HUD and a menu usually need different rules: the HUD can start visible, while the menu waits for a button, action or script.

Show GUI When	Typical use
Visible at Start	A persistent HUD or opening overlay.
Called by Trigger Action / Visual Script	A menu, notebook or panel shown explicitly through an action.
A Mouse Hover screen zone	A toolbar that appears when the pointer reaches an edge, corner or center area.
A screen-wide mouse click	A context-style interface opened from a left/right click or double click.

1. For Adventure HUD, choose **Visible at Start** and click **Apply Activation**.
2. For Workshop Menu, choose **Called by Trigger Action / Visual Script** and apply it.
3. Save. The HUD should be available in play; the menu should appear when Menu calls it.

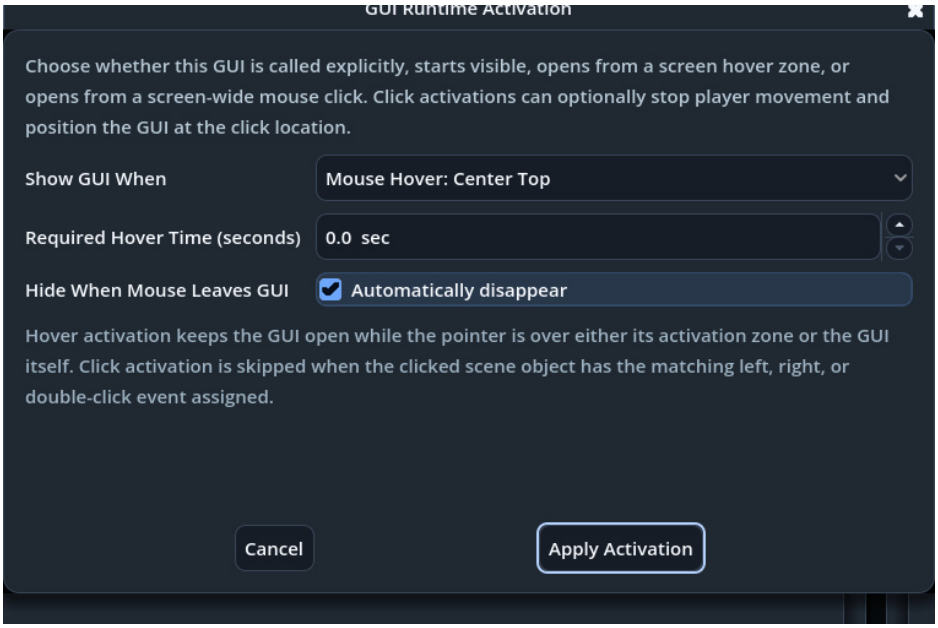

Figure 25. Hover modes add a required hover time and an option to hide when the pointer leaves the GUI.

For a hover toolbar, choose a zone, set a small delay if needed and decide whether it should hide on leaving. It remains open while the pointer is over the activation zone or the GUI itself.

You can also edit Runtime Activation on the whole-GUI Inspector form. Use that form's Apply Changes there; the toolbar dialog uses Apply Activation.

Click activation and deliberate menu calls

Click activation watches the game screen for the chosen click type. It is useful for a contextual verb menu, but it needs to fit the way you already use world clicks.

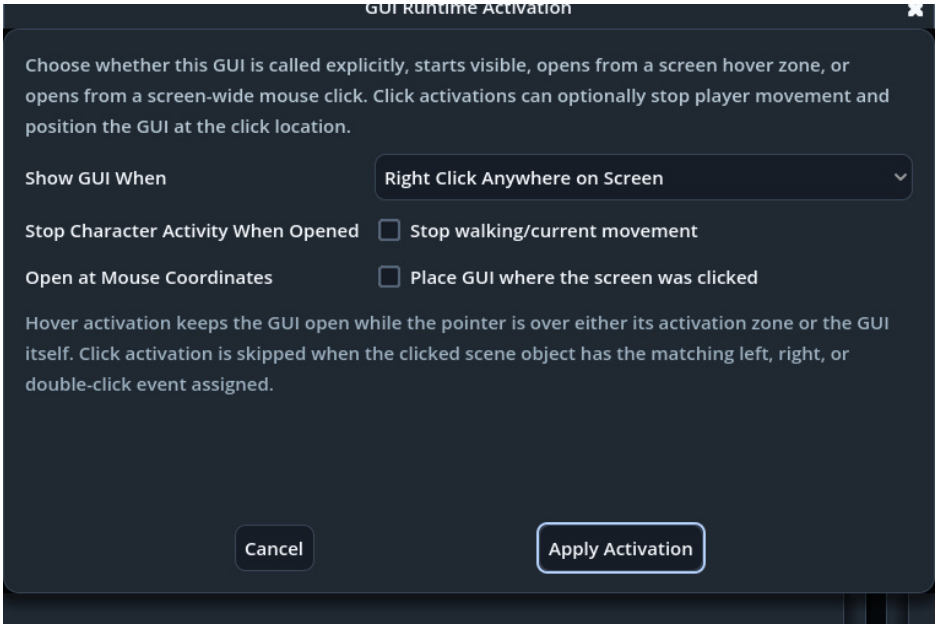

Figure 26. A click-activated GUI can stop the character's current movement and open at the clicked coordinates.

Option	What it changes
Stop Character Activity When Opened	Stops current walking/movement when this click activation opens the GUI.
Open at Mouse Coordinates	Places the small GUI at the clicked screen point. It is not a way to relocate a full-screen HUD.

Matching scene-object click events can take priority, so a screen-wide opener should not be assumed to override every authored object action. Test clicks on empty floor, on an interactable object and over an existing GUI.

Use an explicit call when it is clearer

For an ordinary Menu button, keep the menu trigger-controlled and use Show GUI. Use Hide GUI for the close/return path and Toggle GUI only when the same control should open or close it according to its current state.

> Showing and hiding are different from enabling and disabling. A disabled GUI or element can remain visible. Use the matching action for the change you actually want.

15. Set opening and closing effects

Click **Transitions** for the selected GUI. The two sections are independent: When GUI Opens is used for showing; When GUI Disappears is used for hiding.

1. For Workshop Menu, choose **Fade** in When GUI Opens and set Duration to `0.30` seconds.
2. Choose **Slide** in When GUI Disappears and set Duration to `0.25` seconds.
3. Choose a direction appropriate to the closing movement.
4. Apply the transitions, save and test opening and closing the menu in play.

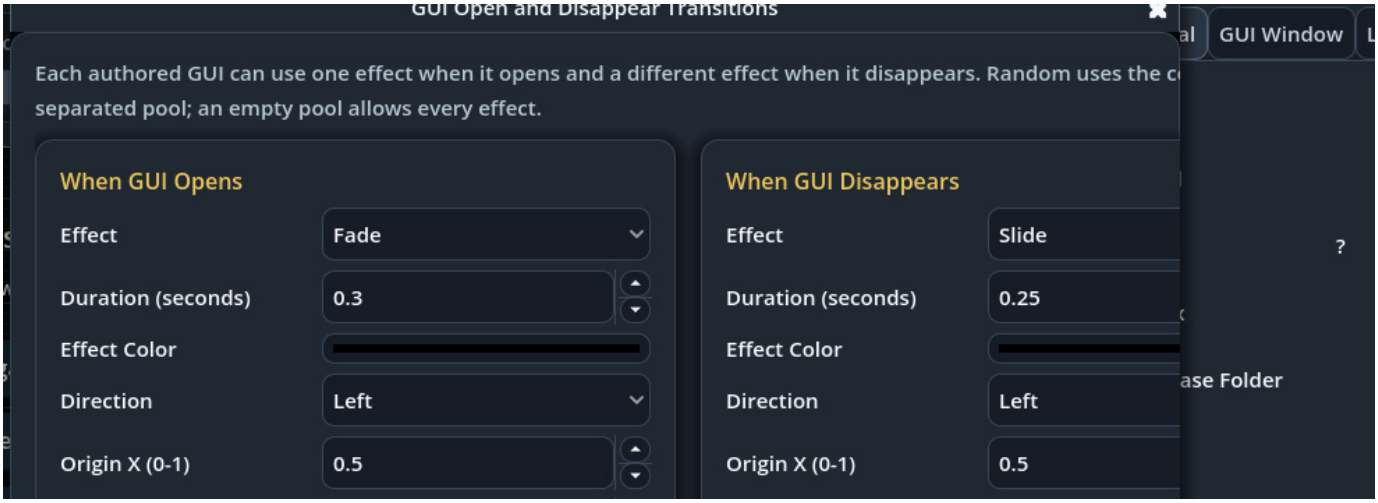

Figure 27. The menu uses different open and close effects. The runtime test opens with Fade and closes with Slide.

Setting	How to read it
Effect / Duration	The transition and how long it takes. Cut is immediate.
Effect Color / Direction	Appearance and movement options used by the chosen effect.
Origin X / Y	A normalized focal point: 0 to 1, with 0.5 at the center.
Intensity	The effect strength.
Random Pool / Avoid Immediate Repeat	Limit Random to chosen supported effect IDs and reduce back-to-back repeats.

Apply an effect to an element

Transitions change how a GUI appears or disappears. **Element Shader** changes the appearance of selected controls while they are displayed.

1. Select the element, or Ctrl-select several elements that should receive the same effect settings.
2. Click **Element Shader**, or **Edit Element Shader Stack...** in Visual Style.
3. Choose a preset in Layer 1 and begin with a modest intensity.
4. Leave unused layers as None. Click **Apply to Selected Element(s)**.
5. Check the result in Designer Preview and gameplay, then save the project.

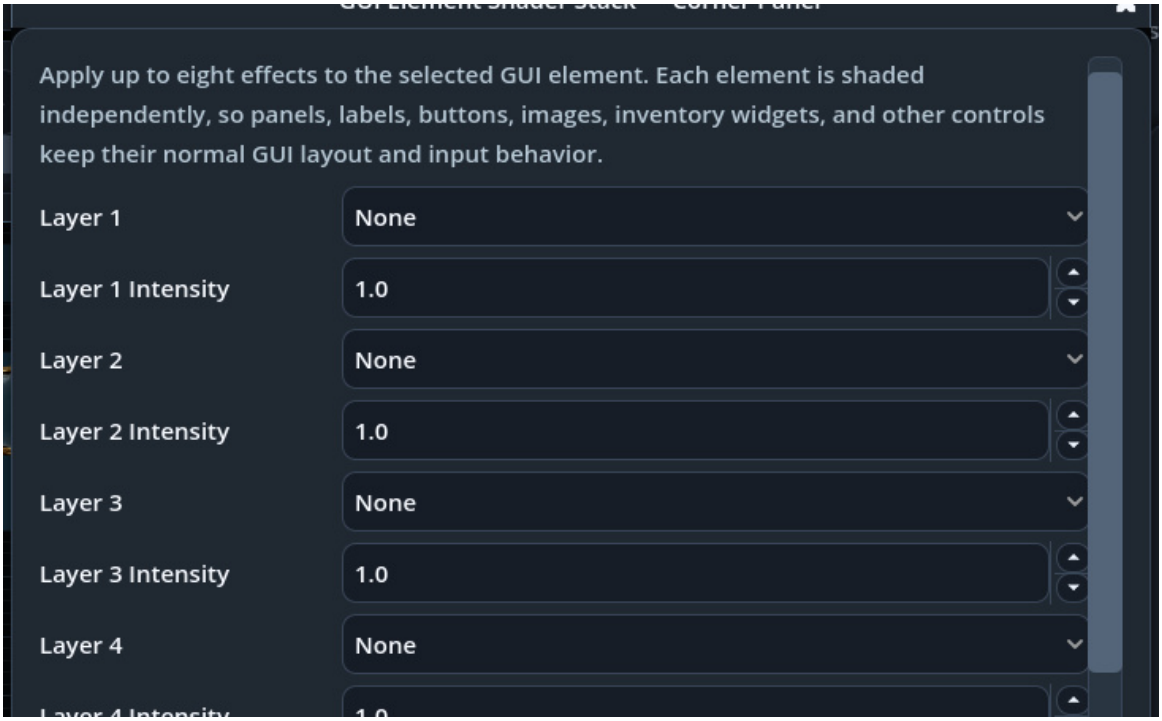

Figure 28. The shader dialog provides up to eight layers. Its controls belong to GUI elements, not to the room's screen-shader stack.

The stack also exposes shared parameters such as pixel/grain/stroke size, distortion amount, animation speed, color/art levels, edge strength and tint. Which ones matter depends on the selected effects.

Use an empty or None stack to remove the selected element's effect. Check all eight layers if an unexpected tint, distortion or glow remains. A multi-selection receives the chosen settings; it is not a blended average of the old stacks.

> The effect does not rewrite the control's layout or its event logic. Keep text readable, and do not rely on a distorted visual edge to redefine the clickable frame. Room screen effects are edited separately.

16. Make a panel with linked GUI tabs

A tabbed Panel can switch between other authored GUIs. This keeps each page manageable: one GUI for notes, another for controls, and a small host panel that chooses which page to show.

1. Create two content GUIs with unique IDs, such as `notes_page` and `controls_page`. Keep them trigger-controlled rather than independent opening HUDs.
2. Create a host GUI, add a **Panel** and open its **Type Settings**.
3. Enable **Tabbed Panel**.
4. In **Tabs (id|Name|GUI ID, ...)**, enter the two records shown below.
5. Set **Editor Active Tab** to `notes`, style the tab text and colors, save and test the host in gameplay.

```
notes|Notes|notes_page, controls|Controls|controls_page
```

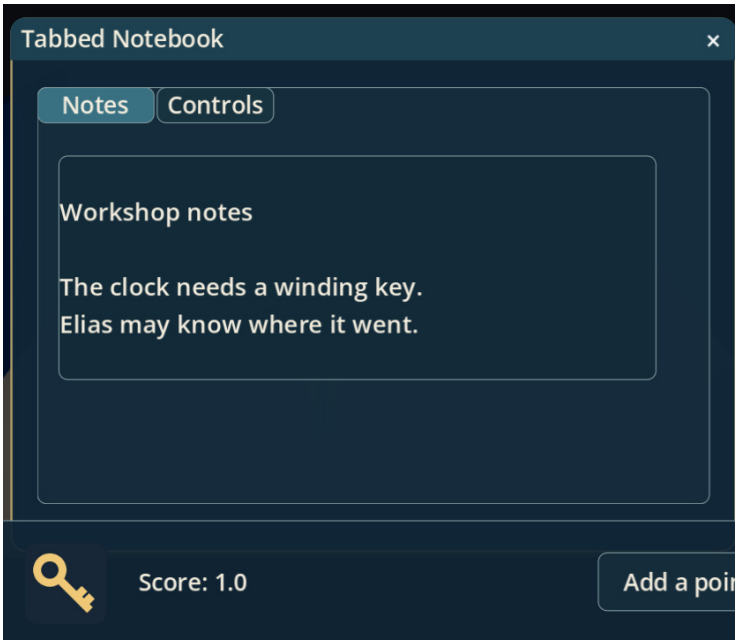

Figure 29. The host notebook showing the linked notes page during gameplay. The tab name and the linked GUI ID have different jobs.

Each entry has a local tab ID, the visible tab name and the GUI ID of its page. Active Tab Name and the active-tab font controls style the selected tab; the default tab controls supply the common style.

> Designer Preview can show an Embedded GUI hint instead of drawing the full linked page. Use gameplay preview to check the actual content. The Legacy Parent Panel ID and Legacy Panel Tab ID fields support older child-element layouts; they are not required for this linked-GUI method.

Use advanced record tools deliberately

Custom properties belong to one record

Clear the element selection and open the whole GUI's **Custom Properties** tab. Add Custom Property creates a row with an ID, display name, type and value. Types include String, Boolean, Number and Array, with a separate blank/null choice.

For example, a String property called page_kind can hold notebook. Apply the GUI record form and save. A Change GUI Property action can then use page_kind as its Property. This is record-specific data, rather than a new global variable in Variable Designer.

Check which Raw JSON you opened

Selected item	Raw JSON represents
A GUI record / GUI Window	The complete GUI, including windowFrame and elements.
A child element	Only that element. Its button is Validate and Apply Element JSON (Advanced).

Change a small, known property and use the matching Validate and Apply button. Keep the surrounding JSON valid. Do not paste a single element over a whole GUI record, or remove its elements array while changing a window setting.

Optional modal input behavior

An authored GUI can have the record property modal set to true. This makes it block gameplay interaction while visible, including its closing effect. It does not mean every world animation and timer is paused.

```
"modal": true
```

If modal already appears in the GUI General form, edit it there. A new record may not expose that field yet; advanced users can add the property to the whole GUI's JSON, alongside its id and name, then validate and apply it. Include commas as required by the surrounding JSON.

For unusual main-window corners or a backdrop not exposed in the ordinary window form, the windowFrame record supports the appearance keys used by property actions. The next chapter shows the action approach, which is easier to test without editing a large JSON record.

17. Change a GUI from a script

Use Show GUI and Hide GUI when you only need visibility. Use Change GUI Element Property for a child control, and Change GUI Property for the window or a GUI custom property.

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

func run(_event: Dictionary = {}) -> Variant:
    await ChangeGUIElementProperty({
        "guiId": "adventure_hud",
        "elementId": "hint_label",
        "property": "text",
        "value": "Ready to explore"
    })
    await ShowGUI({"guiId": "workshop_menu"})
    return true
```

1. Create a GDScript record using this source, with an ID such as `update_hud`.
2. Validate and apply the script, then save the project.
3. Call it from the desired GUI event using a script-running behavior/action, or from an existing action list.
4. In gameplay, check that Hint Label changes and Workshop Menu opens.

Task	Action family
Show/hide a whole interface	Show GUI / Hide GUI
Show/hide one control or group	Show GUI Element / Hide GUI Element
Make a visible control unavailable	Disable GUI Element; use Enable GUI Element to restore it.
Change text, images, colors or geometry	Change GUI Element Property
Change a window appearance/custom value	Change GUI Property

Runtime property changes affect the running interface. They do not rewrite the saved authoring layout. Keep starting values in the designer and use runtime actions for changes that happen during play.

Use visual nodes and property actions

1. Create a visual script, such as Open Workshop Menu(open_menu).
2. Use an On Interact entry and add **Show GUI**.
3. Select Workshop Menu(workshop_menu) in the GUI parameter.
4. Connect the entry's NEXT output to Show GUI's FLOW input.
5. Call this script from the intended interaction, save and test it in play.

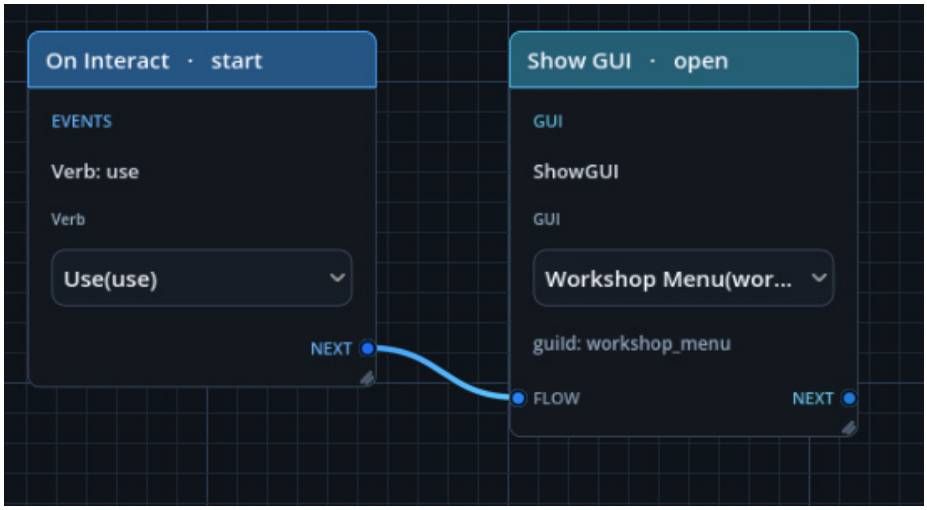

Figure 30. The visual-node counterpart to the same menu-opening action. The connecting line is what makes the next node run.

The same property action in an event or a node

Choose Change GUI Element Property, select the GUI and element, then choose Property and supply To. For example, changing state_button's radiusTopLeft to 32 changes that one corner during play.

Target	Property	Example value
A child element	text	Ready to explore
A child element	hoverImageAssetId	An imported image asset ID
A child element	radiusTopLeft	32
A whole GUI window	windowBorderRadius	14
A whole GUI window	radiusTopLeft	32

Use explicit GUI and element IDs for these examples. The event also supplies caller context such as guild and guiElementId, but an explicit target keeps the setup clear and avoids context-dependent validation problems in this release. Call Script Function is available when you have declared reusable functions; its declared parameters appear for the selected function.

Open the same menu in other languages

These short alternatives all call the same shared GUI action. You only need the language you are using for your project. Create a record in the matching script designer, validate/apply it and call it from an event or action list.

PCA Script

```
language 1.0

function run(context)
    show gui workshop_menu
    return true
end
```

Lua

```
local Script = {}
function Script.run(context)
    await(ShowGUI({guiId = "workshop_menu"}))
    return true
end
return Script
```

BASIC

```
OPTION PCA3D BASIC 1.0
FUNCTION Run(context AS Dictionary)
    AWAIT ShowGUI({"guiId": "workshop_menu"})
    RETURN TRUE
END FUNCTION
```

Each version opened the same Workshop Menu in the runtime checks. Use HideGUI / the matching hide-gui statement for a closing path, or call the shared Hide GUI action from the event editor.

Native GDScript also provides GetGUIProperty, SetGUIProperty, GetGUIElementProperty and SetGUIElementProperty for direct reads and writes. Supply the GUI ID, element ID where needed, property key and value/default appropriate to that helper.

18. A first look at text entry

Add **Text Box** when the player should type into the interface. Its starting contents are edited through Text & Content; its input behavior is under Type Settings.

1. Add a Text Box, give it a useful ID and make it wide enough for the expected answer.
2. Open **Type Settings** and enter a Placeholder, such as Type a puzzle answer...
3. Choose single-line or **Multiline** according to the kind of input.
4. For a single-line parser box, enable **Run Text Parser Command When Enter Is Pressed** only when your parser is configured.
5. Decide whether Enter should clear the text and return focus to the box. Save and test in gameplay.

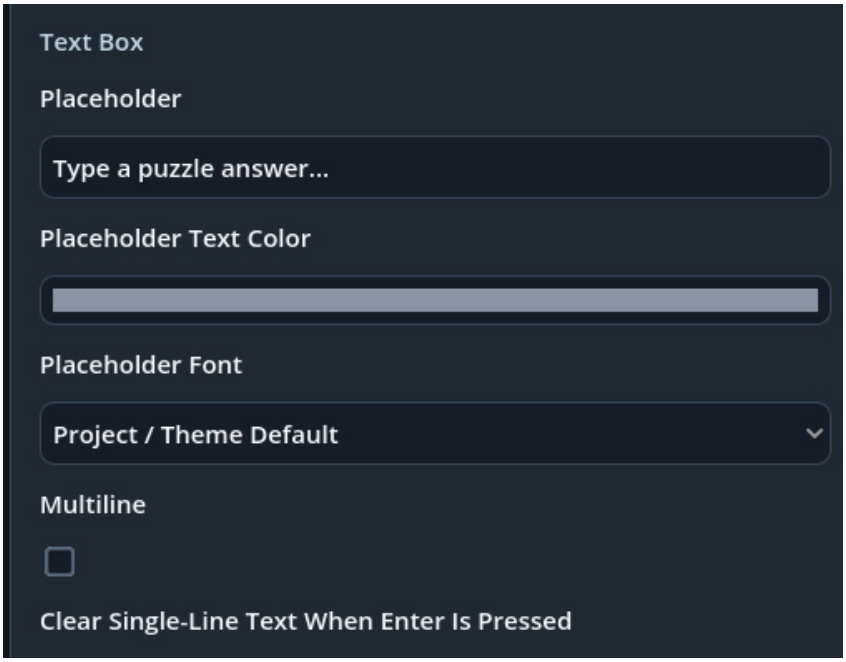

Figure 31. Text Box settings control the entry behavior. Placeholder is guidance shown while the box is empty.

Placeholder text is not an answer the player has entered. A normal text box also does not become a global variable simply because you gave it an ID. Connect the intended game logic explicitly.

Focused text entry blocks game hotkeys, so typing letters such as I or J should not open inventory or journal windows. Test focus on the target devices, including the on-screen keyboard if you support touch input.

Password/Cipher Mode provides visual masking and character replacement options. Those deserve their own input/puzzle tutorial; masking alone should not be treated as secure storage or encryption.

Other controls: the basics

These controls use the same General, Text & Content, Visual Style and Images tabs as the simpler elements. Type Settings supplies their basic behavior. This is an introduction, rather than a full widget setup.

Element	Start with
Check Box	A readable caption and Checked on/off. Use it for an independent yes/no choice.
Radio Button	A caption, Checked state and a shared Radio Group for mutually exclusive options.
Slider	Minimum, Maximum, Value and Step. A 0-100 range with a step of 5 is easy to test.
Progress	Maximum, Preview Value, fill direction and fill appearance. It displays an amount rather than letting the player drag a value.

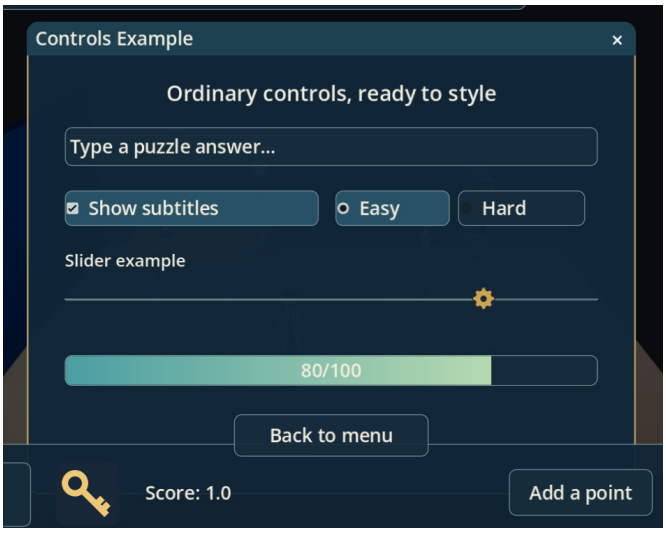

Figure 32. The controls example in gameplay. The slider and progress display are deliberately connected to the same example value.

A slider's draggable handle has separate image choices in **Images > Slider Handle**: Slider Thumb Image and Slider Thumb Hover Image. Visual Style provides thumb size and the fallback color/glow controls.

> A slider captioned Volume does not automatically change the game's audio volume. The control, its data and the setting it should change still need a deliberate connection. Detailed bindings, progress events and control-specific behavior are left for later tutorials.

19. Add a video or a 3D model

Video and 3D Model are media elements inside a GUI, rather than objects placed in the room. Give them enough Width and Height, choose the matching imported content in Images > Media Asset, and test them in play.

Element	Basic setup
Video	Choose a video asset, then set Autoplay, Loop, Muted and Volume in Type Settings. A Poster Image and Load Failure Text provide useful fallbacks.
3D Model	Choose a model, select an Initial View and use Center & Fit Model if it is poorly framed. Starting Camera Distance 0 enables automatic fitting.

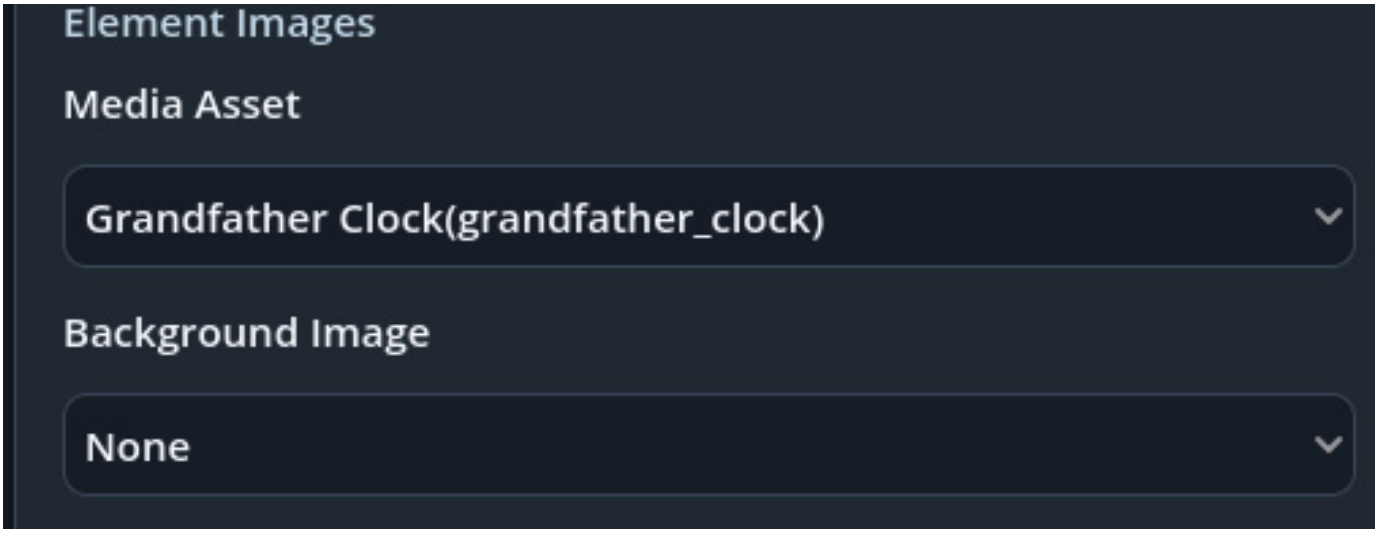

Figure 33. Choose the model content in Media Asset. A separate Background Image can remain behind the viewer.

A 3D viewer can allow player orbit, pan and zoom independently. Its animation options use clips present in the chosen model. Start with the default view before adding animation and input options.

A video has its own content-fit and margin settings. A model has view, rotation and framing settings. The surrounding GUI surface is styled separately from the rendered content.

> Keep this first test small: load one asset, fit it and verify it on the target platform. Model inspection controls, animation workflows and detailed video playback will be covered separately.

20. Recognise the data-driven widgets

Widgets combine several controls and read game data. Add them with Add Element and place their outer frame as you would any other element. Their Type Settings and internal styles are more extensive than a label or button.

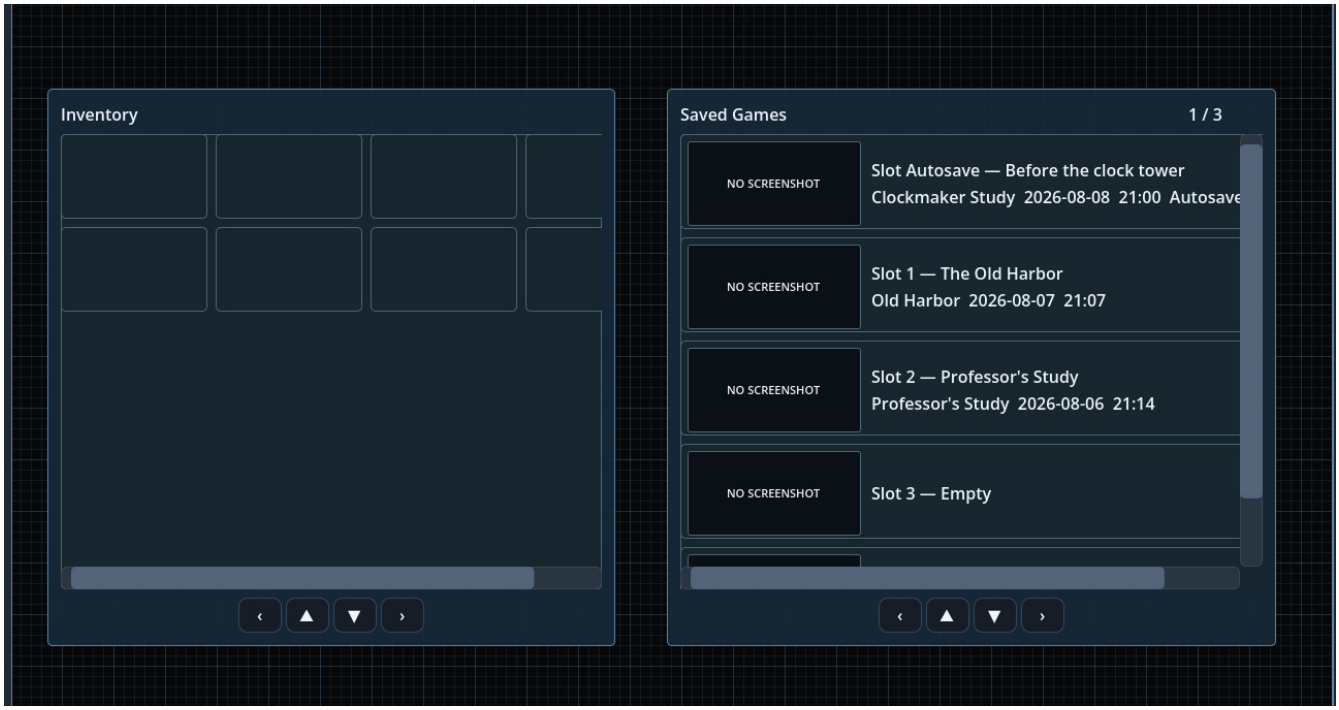

Figure 34. An Inventory Grid and Save Game Widget in the designer. The save rows are editor sample content; they are not a test of real saved-game files.

Widget	What it is for
Inventory Grid	Shows carried inventory in slots, with paging and item interactions.
Save Game Widget	Presents save slots and supports the chosen save/load/browse role.
Journal List	Shows journal/quest information rather than a manually typed list of labels.
Hint List	Presents the game's configured hint content.

For a first pass, choose the appropriate type, size it, select any required source/role and check it in gameplay. Do not try to reproduce the widget's internal behavior by stacking a large number of ordinary buttons on top of it.

Inventory/save widgets have several surfaces: outer widget, grid area, slot frame and, for save widgets, screenshot area. Their corner and image settings can be independent. The dedicated tutorials will explain those controls in detail.

Lists and carousels: a brief orientation

Widget	What it displays	Basic starting decision
List Box	Selectable rows.	Choose manual items, an Array variable or inventory as its source.
Image Carousel	A sequence of imported images.	Choose its initial image assets.
Inventory Image Carousel	Inventory items presented as image cards.	Choose the intended inventory owner/category.

The list box can report the selected item through a selection variable. A carousel can expose arrows, touch/swipe, keyboard or wheel navigation. Those are control-specific choices, not settings you need to enable on every GUI.

Keep the first check modest

1. Put one widget on a small test interface in your existing project.
2. Give it a stable ID and enough space for its title, contents and navigation.
3. Check the source or role it should use.
4. Run a gameplay preview with actual data: a carried item, a saved slot or a populated list.

Editor samples help you arrange a layout. They do not prove that the runtime inventory, save folder, quest data or Array source is populated. Treat a data check as a separate step from an appearance check.

> This guide stops at identifying and placing the widgets. Their detailed sources, slot/row layouts, internal images, pagination, filtering and event context will be handled in the separate tutorials.

21. Assign GUIs to the game's system roles

Showing an arbitrary GUI and assigning a system role are different things. A role tells the game which authored interface to use when it needs its HUD, inventory, dialogue or a standard menu.

1. Open **Global Settings**.
2. In the current 0.86.3.4 layout, use **General** and scroll to **Custom System Interfaces**.
3. Set **Main HUD** to Adventure HUD(adventure_hud).
4. Apply the settings form and save. Start a fresh game preview.

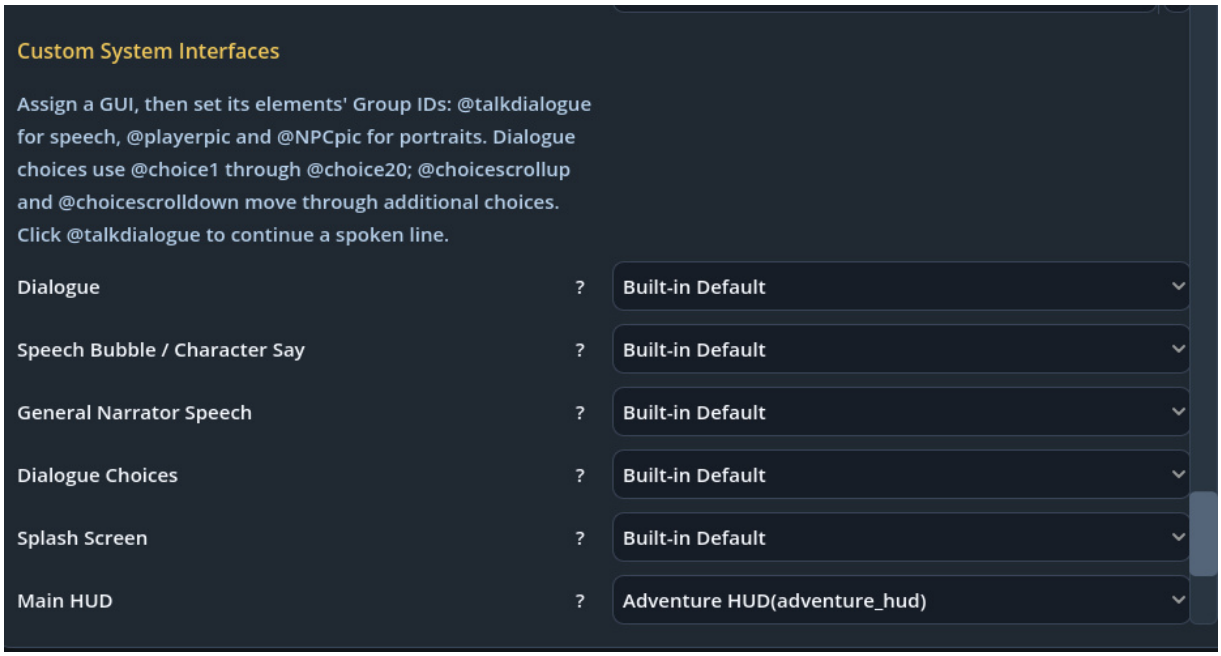

Figure 35. Custom System Interfaces in Global Settings. Main HUD is assigned to the authored Adventure HUD record.

Role	Use an authored GUI when...
Main HUD	You want your own gameplay controls and information display.
Inventory / Journal Screen	You want the corresponding data widgets inside a custom interface.
Pause / Main Menu	You are replacing a standard menu presentation.
Save / Load / Options Screen	You have built the controls needed for that system function.

A blank assignment uses the built-in default where provided. Main HUD also offers No HUD. A button with a system-options behavior opens the assigned options surface; a Show GUI action opens the specific GUI you select.

Connect dialogue text, portraits and choices

Global Settings also provides Dialogue, Speech Bubble / Character Say, General Narrator Speech and Dialogue Choices assignments. A custom dialogue GUI needs the expected element Group IDs so the game knows where to put its data.

Group ID	Suitable element	What the game supplies
@talkdialogue	Label or button	The spoken/narrator line. Clicking the spoken line can continue it.
@playerpic	Image	The active player character's portrait.
@NPCpic	Image	The speaking character's portrait. Keep the capitalization as shown.
@choice1 to @choice20	Buttons or clickable labels	The available dialogue choices, in authored order.
@choicescrollup / @choicescrolldown	Buttons	Movement through additional choices.

1. Create the dialogue GUI and add a generously sized line element.
2. Set that element's **Group ID** to `@talkdialogue`.
3. Add portrait Image elements with the appropriate portrait Group IDs if needed.
4. Assign the GUI to the relevant dialogue/speech/narrator role, then test a real conversation.

Ordinary Group IDs are for your own grouping. These @ names are special runtime connections. Do not use them as generic grouping names for unrelated buttons.

Portrait sources, animation and lip sync are configured through the character/portrait workflow. Putting an Image on a GUI does not by itself create lip sync. Dialogue timing and available choices still come from the dialogue setup.

> Reserve enough space for translated lines and more than one choice. If you only draw two choice controls, do not assume every longer dialogue will be readable without a paging/scroll plan.

22. Test the complete HUD-to-menu flow

Now test the interface as a player would use it. Start a fresh game preview after saving, rather than relying on the last appearance state left in Designer Preview.

1. Check that the HUD is visible and the menu is initially hidden.
2. Use Add a point once. Score should change from 0 to 1.
3. Click Menu. Workshop Menu should open with its Fade effect.
4. Click Update objective. The instruction should change and the menu should close with Slide.
5. Open the menu again and use Return to game, then test its close button.
6. Choose Look and Use in the HUD, then interact with an object that has the corresponding verb actions.



Figure 36. The authored menu open over the running room. Its contents and window chrome are separate from the HUD behind it.

For a modal menu, clicks should not operate the room behind it while it is displayed. Keep the test focused: a menu that opens, updates one value and closes is enough to prove the main connection before adding more commands.

Use the preview Debug Console when you need to inspect a variable. A displayed console result is a past query, so submit get again after the action you are checking.

Check size, input and presentation

Try this	What to look for
Normal game size	Text is readable and buttons have comfortable hit areas.
A smaller and larger window	Important controls stay inside the viewport; long captions still fit.
Open, close, reopen	Transitions finish cleanly and the correct GUI remains on top.
Hover, hold, release	The held picture does not remain stuck after release.
Press and release outside the original control	The pressed state clears; focus/input behavior remains usable.
Touch input on a supported target	Required commands do not depend on hover or double-click alone.
Keyboard/controller setup	Your actual bindings and focus route activate the intended controls. Do not infer this from mouse tests.
Another language	Longer text, special characters and chosen fonts still work.
Target-platform build	Imported assets, fonts and media are available where the game will ship.

Keep the useful information visible

Read the interface over a bright room and a dark room. Check the normal, hover and disabled appearances separately. A beautiful translucent panel can still be a poor backdrop for small text.

Keep key actions available through a plain button or tap. Right-click and hover can add convenience, but they should not be the only way to reach an essential command on a device that cannot perform them comfortably.

> Editor zoom and editor UI Size are development conveniences. They do not replace a runtime check at the game's actual design and output sizes.

23. Find the cause when something is missing

Symptom	Check first
The Inspector shows the whole GUI	Select one child element. GUI Window and no-selection show the GUI record instead.
The detailed element fields disappear	You may have selected several elements. Return to a single selection.
A toolbar button seems absent	Scroll the designer toolbar horizontally.
You cannot move or resize a control	Turn Preview off and focus the canvas.
A control is outside the frame	Check Anchor, X/Y, Width/Height and the GUI canvas size.
A background covers the buttons	Check the panel's Z Index and the Arrange order.
The whole interface is behind another	Check GUI Z, not only the child element Z values.
A menu never appears in play	Check its activation, the calling event, exact GUI ID and any system assignment.
The picture is stretched or cropped	Check Content Fit for foreground media and Background Fit for the backdrop.
One corner ignores the overall radius	Look for that corner's explicit override. -1 restores inheritance.

Use the element list to find a hidden or covered control. Do not keep adding another copy because the first one is difficult to select. Inspect the existing element's ID and rectangle first.

> Run Validate after substantial edits. It helps with missing references and geometry, but a valid project can still have a logically wrong menu target or an unreadable layout.

Trace an event without guessing

Symptom	A focused test
The button looks right but does nothing	Use gameplay preview. Then inspect the correct Click/Right Click event and its target.
It does nothing only in Designer Preview	Gameplay events are intentionally not run there.
A value changes twice	Check primary behavior, additional actions, legacy action and another input event.
A copied button closes the wrong GUI	Its stored event target may still name the original interface.
A $$ token remains visible	Check the exact variable ID and that it exists in the running game.
A list/inventory/save area is empty	Check actual runtime data separately from editor sample content.
A linked tab shows only a hint in the designer	Test the host in gameplay and verify the linked GUI ID.
A runtime property change is lost on restart	It changed the running interface, not the saved authoring record.

Reduce the test to one clear result

Temporarily use one button and one simple result: show a known GUI, set a known variable or display a short message. Confirm that step before restoring a longer action sequence.

If you change IDs, inspect all consumers: events, visual nodes, scripts, system roles and tab links. A readable display name helps you select the right record, but it is the stored ID that makes the connection.

The examples in this guide were checked in the running editor: image states, a real mouse-operated score button and menu opener, action-list updates, linked tabs, visual-script calls and runtime property changes. Use the same small checks as you extend your own interface.

24. Finish with a clean working interface

Before moving on to a specialist widget tutorial, make sure the shared GUI tools and the basic game connection are solid.

Check	Expected result
Records and IDs	Each GUI and element is easy to identify; event targets name the intended record.
Canvas and anchors	The HUD fits the design area, and the menu opens in the intended position.
Text and assets	Captions are readable; imported images and fonts resolve correctly.
Normal/hover/clicked images	The right artwork appears and the held picture clears after release.
HUD value display	The score and objective tokens resolve to live values.
Menu actions	Open, update and close work in gameplay, in the intended order.
Layers and visibility	Backgrounds stay behind controls; hidden/disabled states are deliberate.
System roles	The game uses the authored HUD/dialogue/menu where you assigned it.
Validation and saving	Relevant issues are reviewed and the accepted authoring work is saved.

Where to go next

Keep the first interface straightforward. A clear objective, a few verbs and a menu that reliably opens and closes do more for a playable adventure than a crowded screen full of unfinished controls.

From here, the separate widget guides can build on the same skills: stable IDs, deliberate source data, independent surfaces and a gameplay test that proves the intended action.

> Save the project, stop the preview and reopen a familiar GUI once more. Check that the starting layout and event definitions are still the ones you meant to keep.