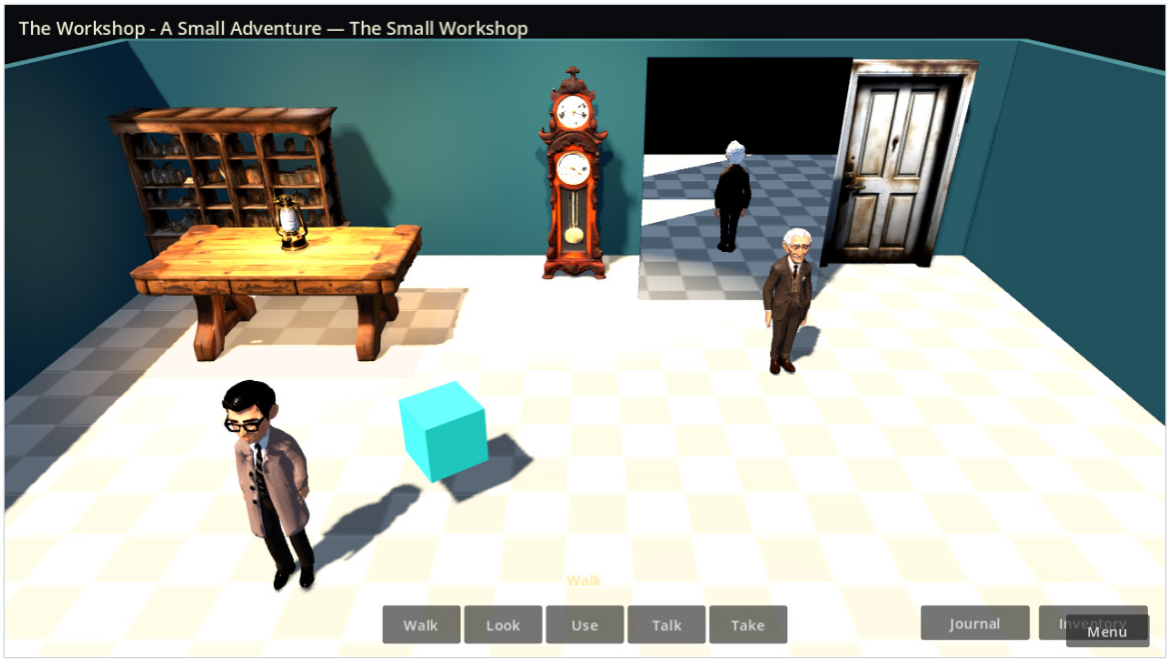
PCA 3D ADVENTURE STUDIO

The Scene Designer Part 3

Finishing your scenes



Make the scene ready to play.

Cameras, atmosphere, reflections, sound, attachments, room changes, and the complete 2D / 2.5D stage workflow. A practical guide to the tools left after the first two Scene Designer tutorials.

For version 0.86.3.4 - Windows, macOS and Linux

Actual editor screenshots. No practice files required.

How to use this guide

This is the final part of the Scene Designer series. Use your own saved workshop or another working scene. Read a chapter when you need its feature, or follow the book in order to make a complete finishing pass.

Chapters	What they cover
1-3	Clearer editing views, organization, navigation and collision.
4-7	Camera bounds, following, advanced shots and focus.
8-10	Lighting, atmosphere, decorative effects and actual mirrors.
11-15	Object effects, motion, interactions, sound, attachments and regions.
16-19	Screen effects, weather, overlays and connected rooms.
20-24	Backgrounds, scrolling, navigation, parallax, walk-behinds and stage objects.
25-28	Scenery generators, advanced data, movie-room connections, actions and final testing.
29	A feature index across all three guides.

The chapters give small, reversible examples. Add one feature, apply it, and test before moving on. Values belong to the pictured example; use the same reasoning with your own scene scale and artwork.

The screenshots were captured from the running editor. Close-ups are cropped to make the controls readable. The main captures use Compatibility rendering; renderer-dependent controls are explained without pretending their effects are visible on an unsupported backend.

Apply, save, then test. No companion project, practice files, or separate asset download is supplied.

Contents

Click a chapter or page number to jump to it. PDF bookmarks include the section headings.

1. Before you add the finishing touches . . . 4
2. Get a clearer editing view . . . 5
3. Shape the walkable area properly . . . 8
4. Limit the camera without limiting the player . . . 13
5. Make an ordinary camera follow the player . . . 14
6. Use Advanced Camera for more deliberate shots . . . 16
7. Focus the shot with depth of field . . . 20
8. Give the room a believable light source . . . 22
9. Add flares, shafts, and decorative light strings . . . 26
10. Make a mirror, not just a shiny object . . . 29
11. Give an object its own surface effect and motion . . . 32
12. Help the player recognize useful objects . . . 35
13. Put sound in the room . . . 38
14. Attach effects and props to moving things . . . 40
15. Use regions for local effects and interactions . . . 45
16. Build a room screen-shader stack . . . 47
17. Weather, outlines, and cinematic grading . . . 49
18. Add a screen overlay and choose the room's view size . . . 53
19. Connect rooms, entry events, and arrival points . . . 56
20. Start a 2D or 2.5D stage room . . . 62
21. Make a stage larger and let its camera scroll . . . 64
22. Add depth with parallax and walk-behind scenery . . . 67
23. Place sprites, billboards, and stage characters . . . 71
24. Finish the stage's interactive parts . . . 75
25. Use the remaining procedural scenery . . . 77
26. Custom properties, Raw JSON, and movie scenes . . . 79
27. Drive scene features with actions or scripts . . . 83
28. Check the scene before calling it finished . . . 85
29. Where every Scene Designer tool is covered . . . 88

1. Before you add the finishing touches

This is the last part of the Scene Designer series. Part 1 built a simple room. Part 2 furnished it, added animated characters, and gave the clock and door something to do. Here we will work through the tools left over: the ones that control how a scene looks, sounds, moves, and connects to the rest of the game.

Use your own saved room. You do not need a special project or a set of practice files. The workshop in the pictures is a working example, not a layout you have to copy exactly. The later 2D chapters use the engine's bundled storybook artwork to make the stage controls easier to see.

The screenshots come from PCA 3D Adventure Studio 0.86.3.4. These instructions apply to Windows, macOS, and Linux. Your theme, panel widths, or file picker may look different. The important things are the selected record, the tab name, and the field label.

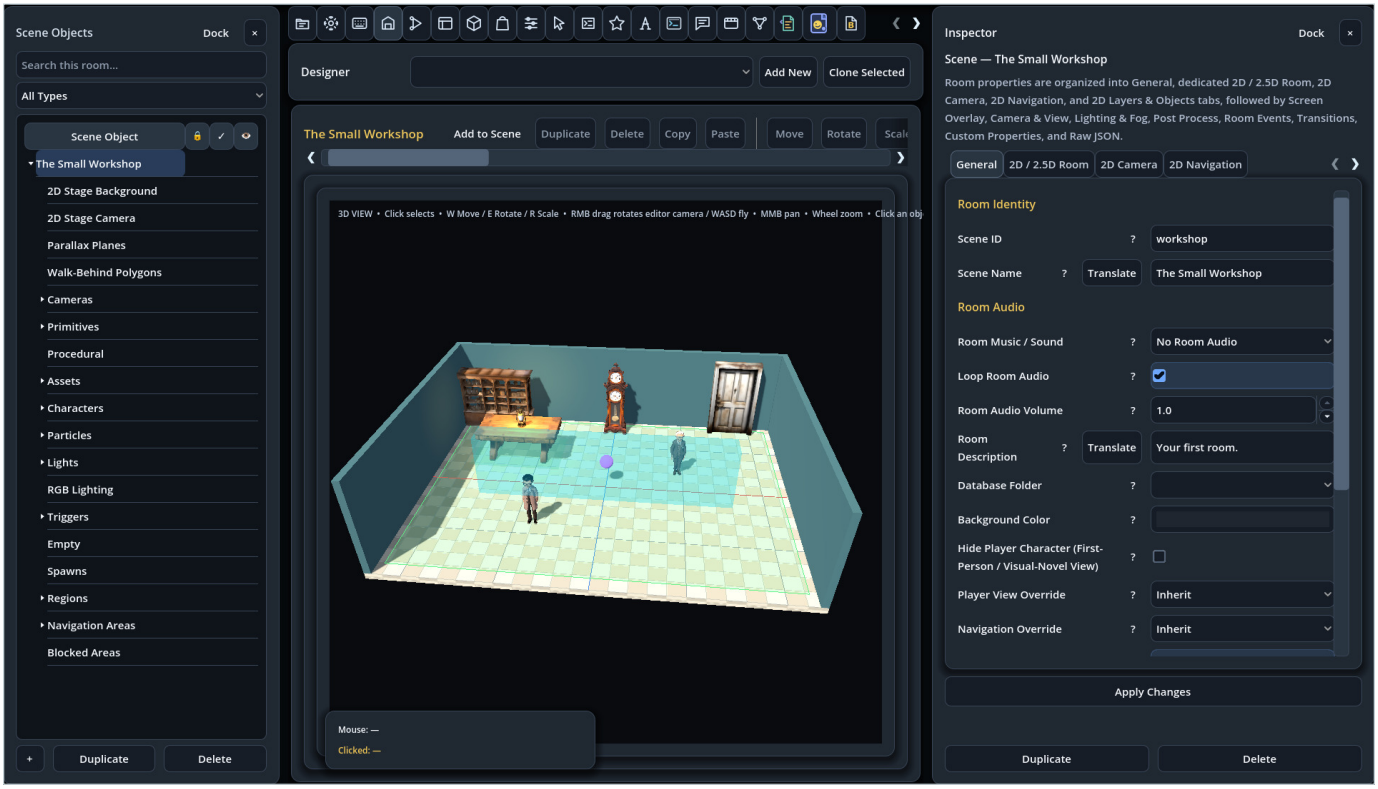


Figure 1. Our starting room. Select the room name for room settings, or an individual item for that item's settings.

Work on one change at a time

1. Open your working project and save a copy with **Save Project As**.
2. Choose the scene you want to improve.
3. Select the correct record in **Scene Objects**. The room itself, a camera, a character, and a light each have different Inspectors.
4. Make one change, click **Apply Changes**, then use **Test Scene**.

5. Click **Stop** before returning to editing. Save the project when the result is right.

Some appearance changes preview immediately. Gameplay, movement, camera tracking, room events, and sound need a running test. Seeing a lamp in the editor does not prove its entry event works.

A few words used throughout the guide

Term	Meaning here
Record	The saved settings for a room, object, camera, or other item.
Instance	One placed copy of an asset or reusable character in a room.
Runtime	The game while it is running in Test Scene or Play / Restart.
Inherit	Use the setting from the room or project instead of supplying a local override.
World unit	A scene measurement. The workshop uses characters roughly 1.8 units tall.
Screen effect	An effect applied to the rendered view, such as grain or a tint.
Material effect	An effect applied to a particular object's surface.

Ordinary **Visual Adjustments** rotation values are in radians. Fields explicitly marked **Degrees**, including continuous rotation speed, use degrees. For reference, 90 degrees is about 1.571 radians. A mismatched unit can turn a small adjustment into a very enthusiastic pirouette.

2. Get a clearer editing view

The view selector changes how you inspect the room. It does not convert a finished 3D room into a 2D room. Room family is a separate choice, covered in Chapter 20.

View	Use it for
3D	Placing and checking the finished geometry from any angle.
Designer	A plan view in a 3D room; a full-stage composite in a stage room.
Mesh	Inspecting the room's geometry without relying on its normal surface appearance.
2D / 2.5D	Inspecting the appropriate flat-room presentation; in a stage room, checking the authored camera crop.
Top, Front, Right, Isometric	Quickly checking a 3D scene along a useful direction.



Figure 2. Designer gives a readable plan of the workshop and its navigation footprint.

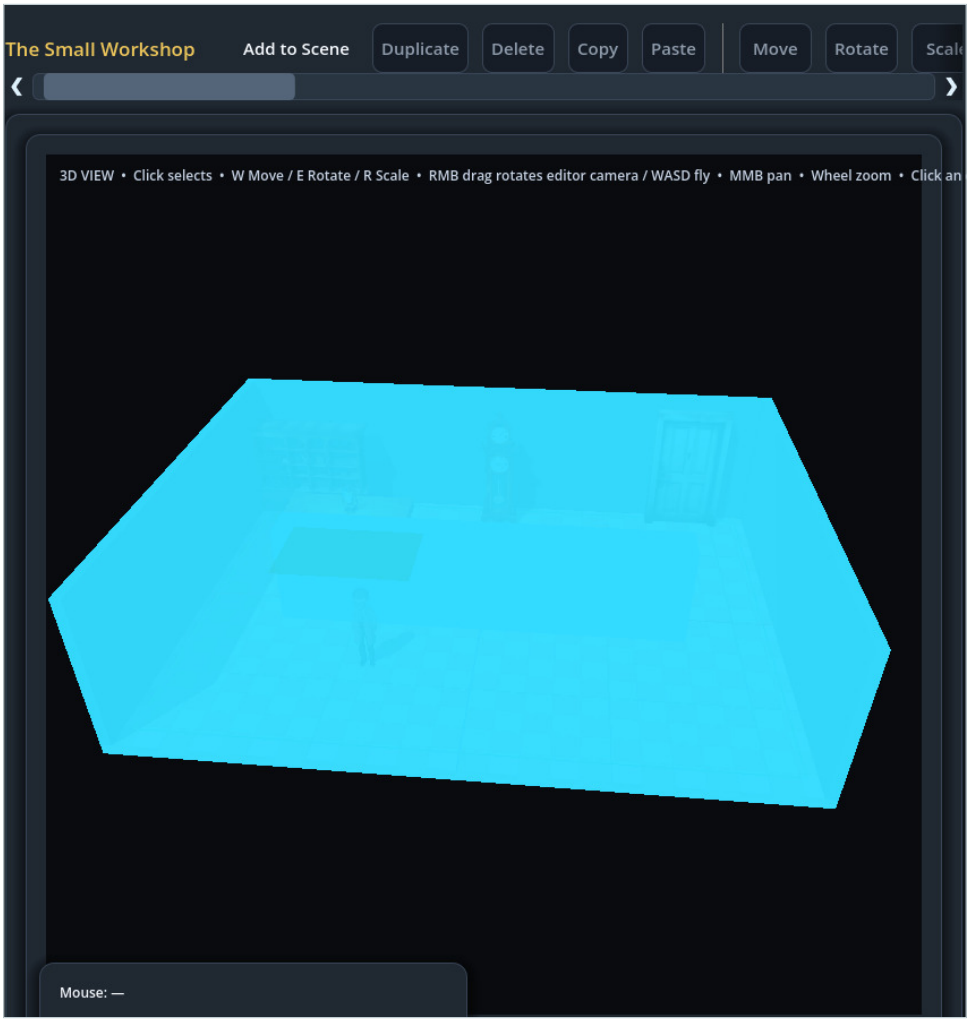


Figure 3. Mesh view is useful when a textured surface makes the underlying geometry hard to judge.

Frame, hide, and lock

Frame Selected brings the selected item into view; **Frame Scene** fits the room. Use these before hunting for a tiny object with the wheel. **F** frames the current selection. The move, rotate, and scale shortcuts are **W**, **E**, and **R** while working in the scene view.

Use **Editor Visibility** to hide categories that clutter the editor. This is different from changing a record's visibility or enabled state. Hiding camera helpers in that menu does not remove the camera from the game. **Clean 2D Room** is another editor-only display aid, explained in Chapter 24.

Lock finished floors, walls, and background art so a drag does not move them accidentally. Unlock an item when you intentionally need to edit its placement. Keep an eye on the padlock rather than repeatedly dragging harder.

Find and organize records

The Scene Objects search box searches the current room. Use its type filter to narrow the list to lights, cameras, characters, or another category. Give each record a readable **Name** and a stable **ID**. A selector such as **Workshop View (wide)** is easier to recognize than `wide` alone, but scripts and references still use the ID.

Duplicate makes a new ordinary object record. Check its new ID, position, collision, and actions before using it. **Copy / Paste** is useful for repeated props. Placed character instances have different rules: clone the reusable character definition when you need a distinct character, then place that character. Do not treat a character instance as another crate to duplicate.

Folders and the **Database Folder / Scene Objects Folder** fields organize records. They do not make a physical parent-child assembly. Moving a folder is not a substitute for an attachment.

Grid controls the visible guide. **Snap** and the adjacent spacing value control placement increments. Neither setting increases how far the player can walk. For that, edit navigation.

3. Shape the walkable area properly

A rectangle is enough for the first workshop. A room with an alcove, a curved path, or a missing corner usually needs a polygon. Navigation tells the player where a route may go; it is not a picture of the floor material.

Turn a rectangle into an editable polygon

1. Select **Designer** or a useful top view.
2. Select the required **Navigation Area** in Scene Objects, or choose it in the **Nav Area** selector.
3. Open the navigation menu and choose **Convert Bounds to Polygon**.
4. Turn on **Edit Navigation** if necessary.
5. Drag the corner handles to match the walkable floor. Keep the outline simple and keep its edges from crossing.
6. Use **Add Vertex** when you need another corner. Select an unwanted vertex and use **Delete Selected Vertex** to remove it.
7. Test routes across the room, along its edges, and around the new corner.



Figure 4. The navigation rectangle has become a polygon with editable corner handles.

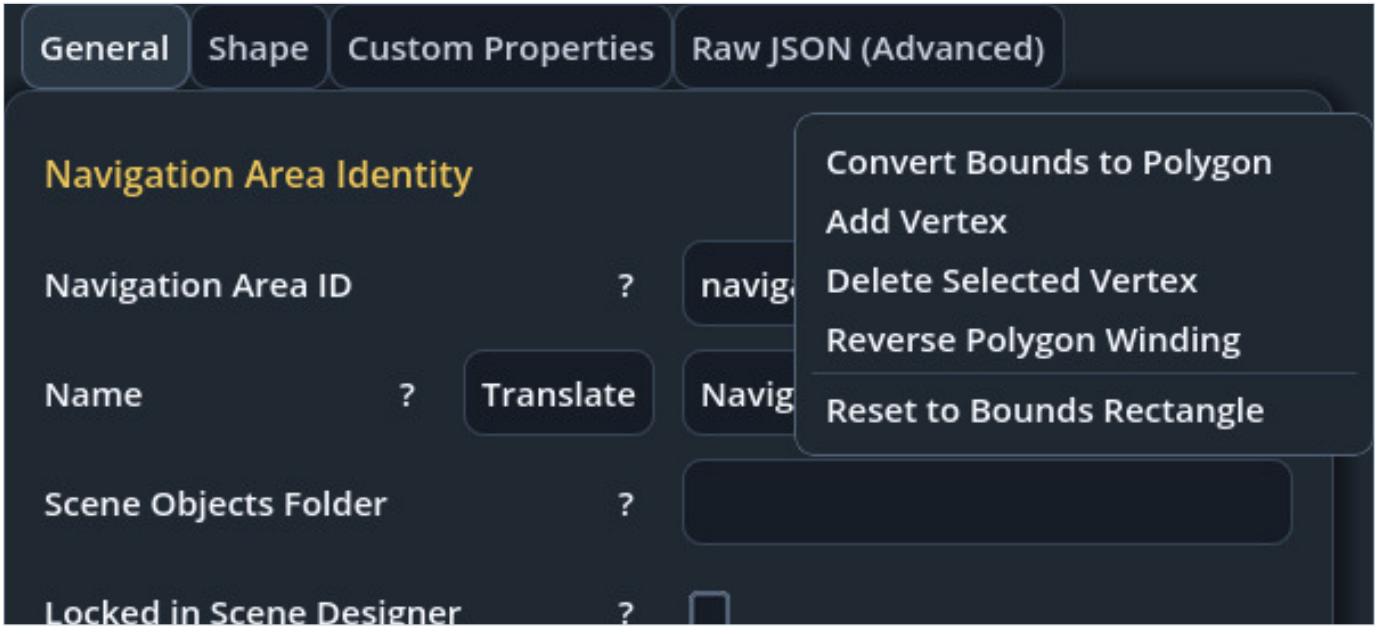


Figure 5. The navigation menu contains conversion, vertex editing, winding, and reset commands.

Reverse Polygon Winding changes the order of the points. It does not enlarge the floor. **Reset to Bounds Rectangle** replaces the custom shape with a rectangle, so use it only when you intend to discard the current outline.

The Navigation Area's **Shape** tab also offers **Bounds Rectangle** and **Editable Polygon**. For a polygon, the vertices are the authoritative shape. Its numerical minimum and maximum bounds are recalculated from them.

More than one navigation area

Use the **+** beside Nav Area to add another area; use **−** to delete the selected area. Enabled, visible areas are combined for navigation. Overlapping areas can form a continuous floor. Separate areas remain separate islands unless there is a valid connection. Drawing a second patch at the far end of a room does not give the character a route across the gap.

Keep characters out of a footprint

Use **Add to Scene > Blocked Area** to exclude a simple area from walking. Set its minimum and maximum X and Z around the footprint that should remain clear. Use an object's physical collision when the character must be stopped by an actual solid object.

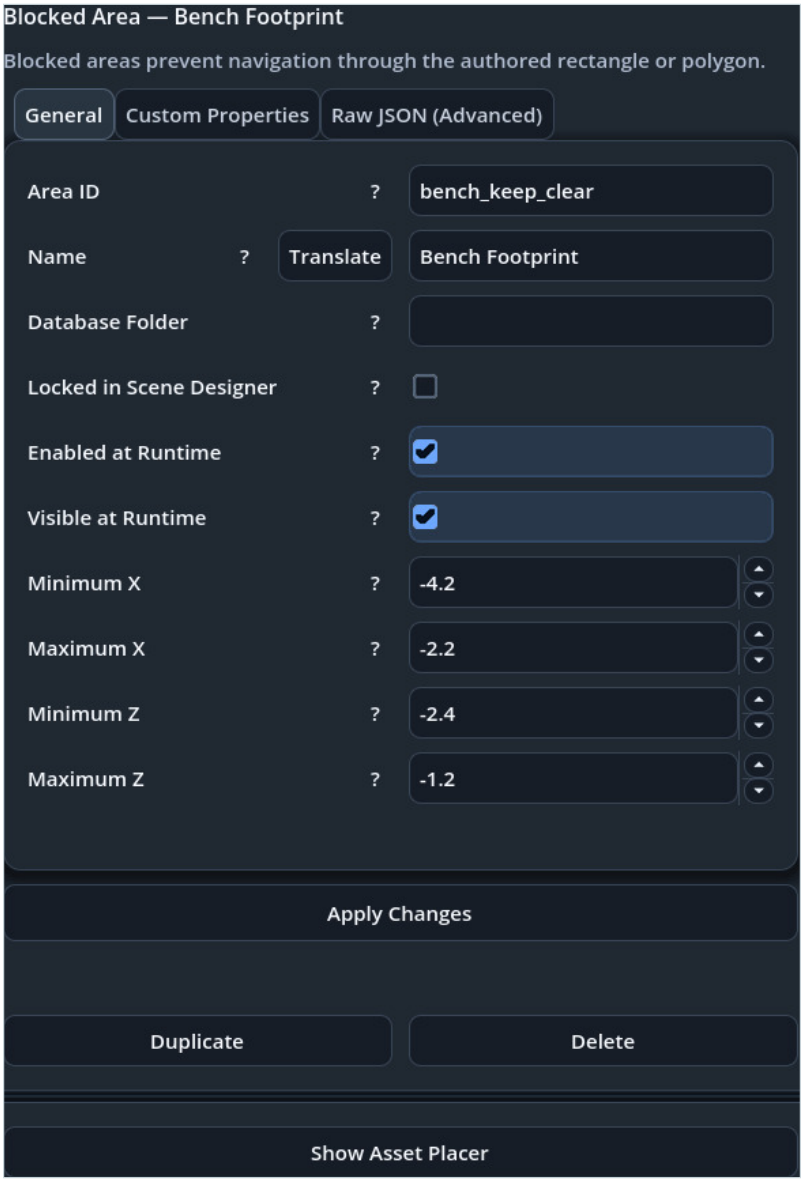


Figure 6. A blocked area keeps the route out of the bench footprint without changing the visible bench.

Do not use an event volume as a wall. An event volume detects entry, exit, or pointer activity; physical blocking and navigation are separate. Likewise, reducing **Stop Distance** changes interaction approach, not the width of a collider.

Raised surfaces and gravity

For an object the player should stand on, enable **Allow Surface Walking on This Object** and provide a suitable physical collision surface. The 3D player must support the relevant movement. A floor-looking texture is not a collision surface, and a large box collider around a detailed staircase is still a box.

The room also stores **Enable Gravity**, **Allow Jumping**, **Gravity**, and **Jump Speed**. In this build those additional room fields are below the entry-transition settings in **Transitions**. Start with normal ground movement, then enable gravity and test the floor, edges, and steps. Jump

permission also needs a usable input or action; ticking the box alone does not teach a point-and-click controller a new button.

The navigation record's ground height, agent radius, and sampling values are advanced data. Use **Raw JSON** for exact values when necessary, after saving a copy. A larger agent radius requires wider passages. Adjust the shape and physical blockers before changing sampling values to solve an ordinary doorway problem.

> **Before you continue**
> The player should reach every intended part of the floor, avoid solid props, and remain out of excluded areas. The grid can extend beyond that floor without affecting any of these rules.

4. Limit the camera without limiting the player

Camera bounds stop the view from wandering beyond the room. Navigation bounds stop the character from walking beyond the floor. They often have similar measurements, but they do different jobs.

1. Select the room name.
2. Open **Camera & View**.
3. Enable **Enable Room Camera Bounds**.
4. Enter the room's left and right X limits, then its top and bottom Z limits.
5. Apply the change and test a following camera near all four edges.

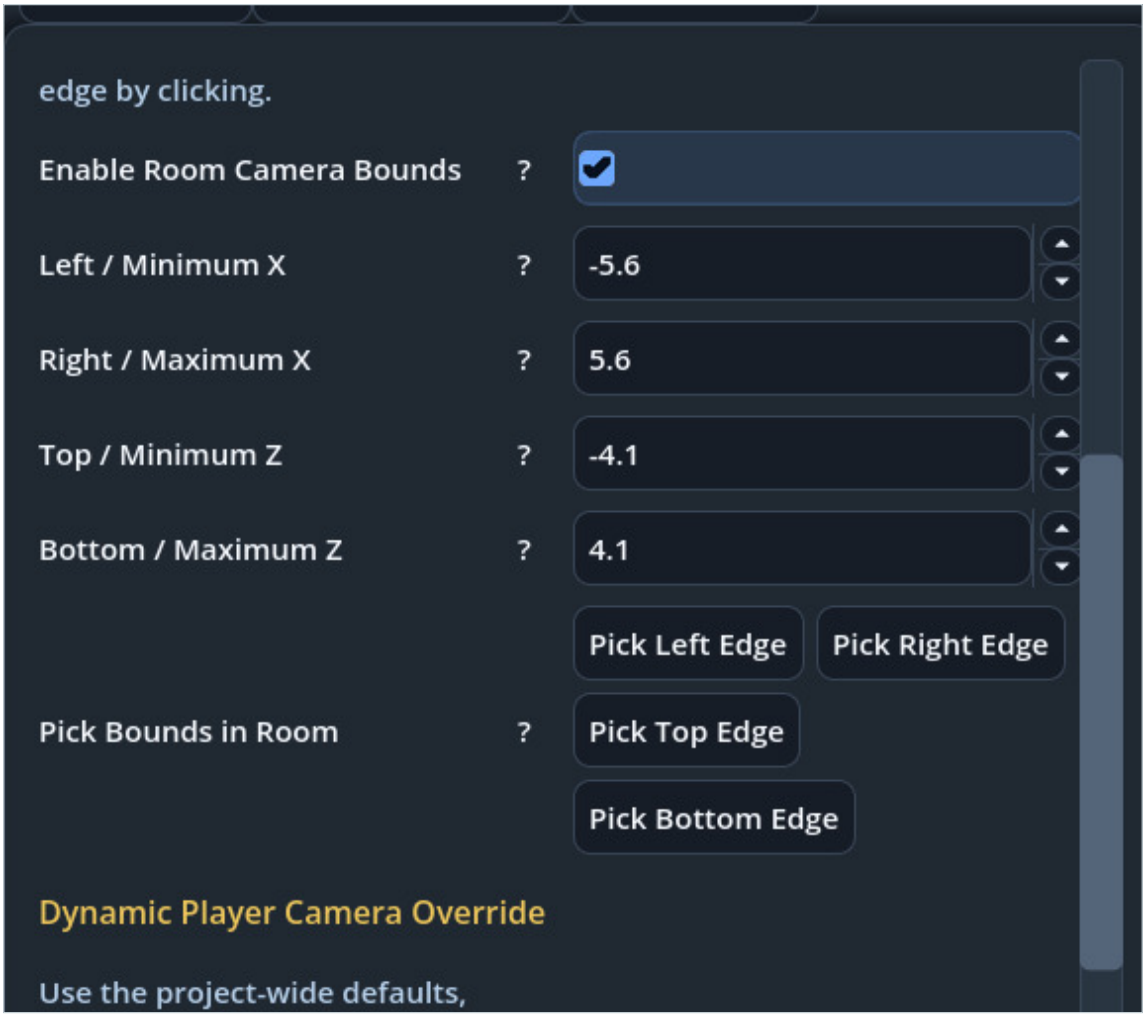


Figure 7. Room camera bounds use world-space X and Z limits.

You can also use the **Camera Bounds** menu in the Scene Designer. Choose an edge, then click that edge in the scene. **Use Navigation Bounds** is a quick starting point. **Disable Camera Bounds** removes the room's camera clamp.

Do not assume that the camera's position should be clamped to the same coordinates as the character's feet. The camera system corrects the shot using its pose. Check the resulting framing in Test Scene, particularly with a steep angle, a wide field of view, or group framing.

Let the player zoom or pan the view

The room's **General** page includes a **Navigation Override** for player view gestures. Despite its name, this section controls camera navigation: **Pinch / Wheel Zoom**, closest and farthest zoom, zoom step, **Two Finger Pan**, and the maximum pan from the centre.

Choose **Inherit** to use the project setting, **Override** to supply room-specific values, or **Disabled** to turn this view control off for the room. Keep zoom limits sensible for the shot; an extreme close zoom can put the view inside a model.

These controls do not change the editor's own orbit and pan controls. They also do not replace the route navigation taught in Chapter 3.

5. Make an ordinary camera follow the player

The **Dynamic Follow** page gives an existing camera simple player tracking. Use this when you like the shot already and just want it to move with the player.

1. Select the camera in Scene Objects.
2. Open **Dynamic Follow**.
3. Set its override to **Use Custom Settings**.
4. Enable **Follow Active Player**.
5. Choose a follow style and the world axes to track.
6. Apply, then walk across the room in Test Scene.

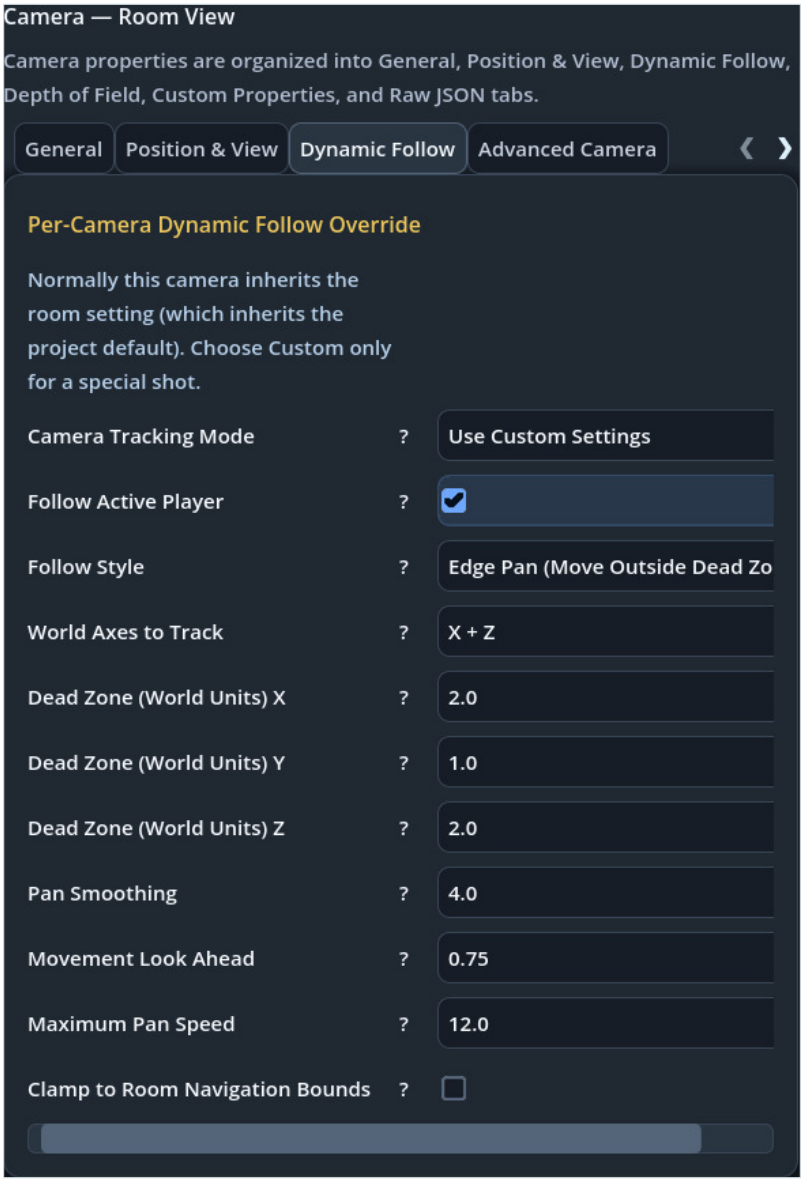


Figure 8. Dynamic Follow adds player tracking to an ordinary camera.

Setting	What you will see
Edge Pan	The camera stays put while the player remains inside the dead zone, then moves when the player leaves it.
Smooth Follow	The camera continuously follows the player.
World Axes to Track	X for side-to-side movement, Z for depth, or the combined axes the scene needs.
Dead Zone	A comfortable area measured in world units, not screen percentages.
Pan Smoothing	How quickly the view catches up. Compare small changes in a running test.
Movement Look Ahead	Shifts the framing in the direction of travel.

Setting	What you will see
Maximum Pan Speed	Caps how fast the camera can chase its target.
Clamp to Room Navigation Bounds	Adds a navigation-based limit; the margin helps keep the framing inside it.

The project supplies the broad default, the room can override it, and the selected camera can supply its own override. If a camera behaves differently from the room setting, check that camera first.

For the workshop, start with **Edge Pan**, X + Z tracking, and a modest dead zone. Walk from one side to the other, stop, and reverse direction. Make sure the camera settles instead of making every small adjustment feel like a boat ride.

Use one camera-follow system at a time while learning. For the next chapter, disable ordinary dynamic tracking on the example camera before enabling Advanced Camera. This makes the source of the movement easy to understand.

6. Use Advanced Camera for more deliberate shots

Advanced Camera is the camera system built around Phantom Camera. You still create and aim a PCA camera in the normal way. Its authored position and target establish the shot's direction and distance; Advanced Camera decides how that shot follows its subjects.

1. Select your wide camera and open **Advanced Camera**.
2. Tick **Enable Advanced Camera**.
3. Choose a **Camera Behavior**.
4. Select the **Follow Target**. For our example, use the player.
5. Apply and use **Test Scene**. The editor's free editing view is not a live demonstration of following.

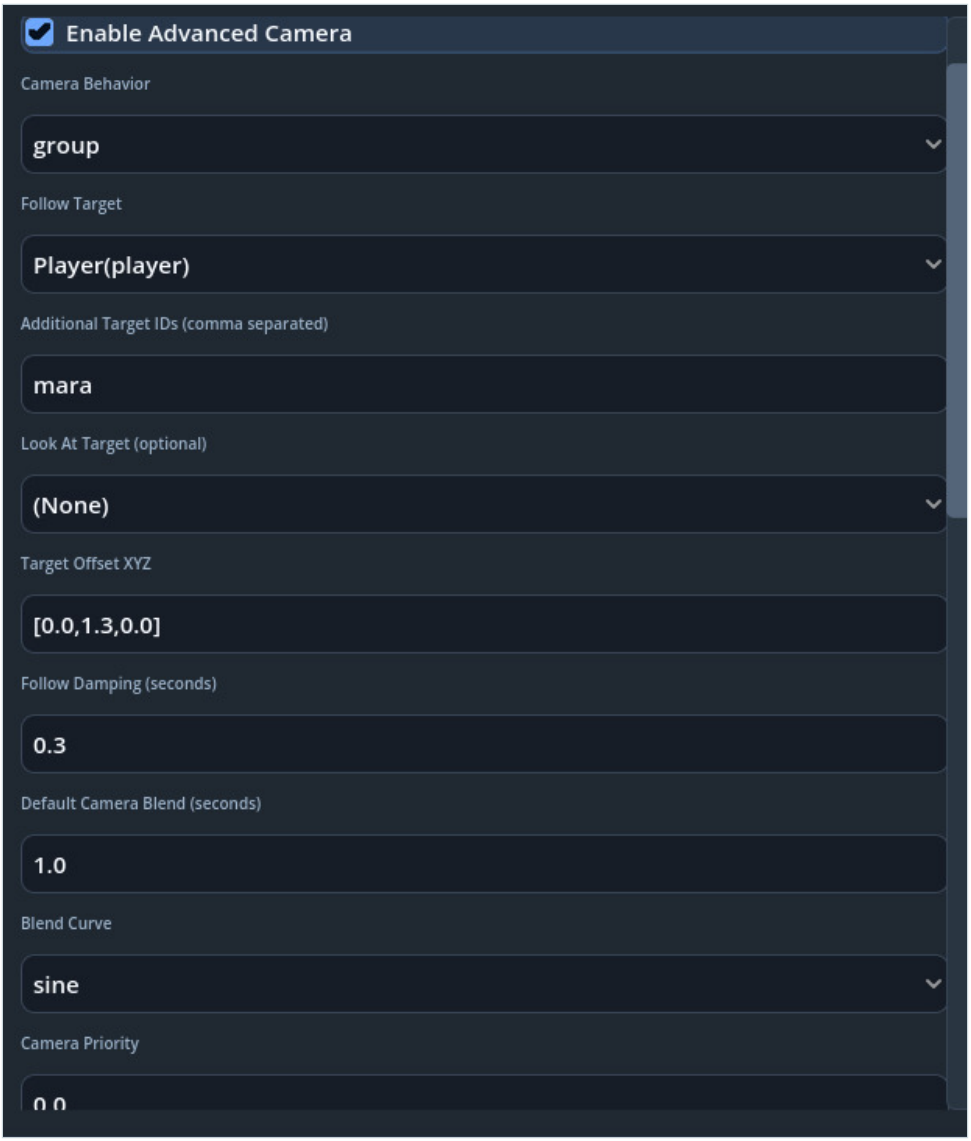


Figure 9. The advanced settings are stored on the selected camera, not on the player.

Choose the behavior that fits the scene

Behavior	Good use
Fixed	A deliberate stationary shot, with the advanced blend controls available.
Follow	Track one subject while keeping the authored shot relationship.
Group	Keep the player and one or more other subjects in view together.
Framed	Leave the shot still until the subject reaches the edge of a framing zone.
Path	Keep the following camera on an authored route.

A conversation shot with Group

Set **Camera Behavior** to **Group**, Follow Target to the player, and **Additional Target IDs** to the NPC's ID, `mara` in this example. This extra-target field expects IDs separated by commas. It does not expect the displayed `Name (ID)` text.

Enable **Automatically Frame Group**. Start with minimum distance 6, maximum distance 18, and padding 1.3 for this workshop. These are example measurements, not universal camera settings. Walk toward and away from the NPC. The view should adjust to keep both characters comfortably framed. If the group is still cropped, allow more maximum distance or reduce the camera's tightness.

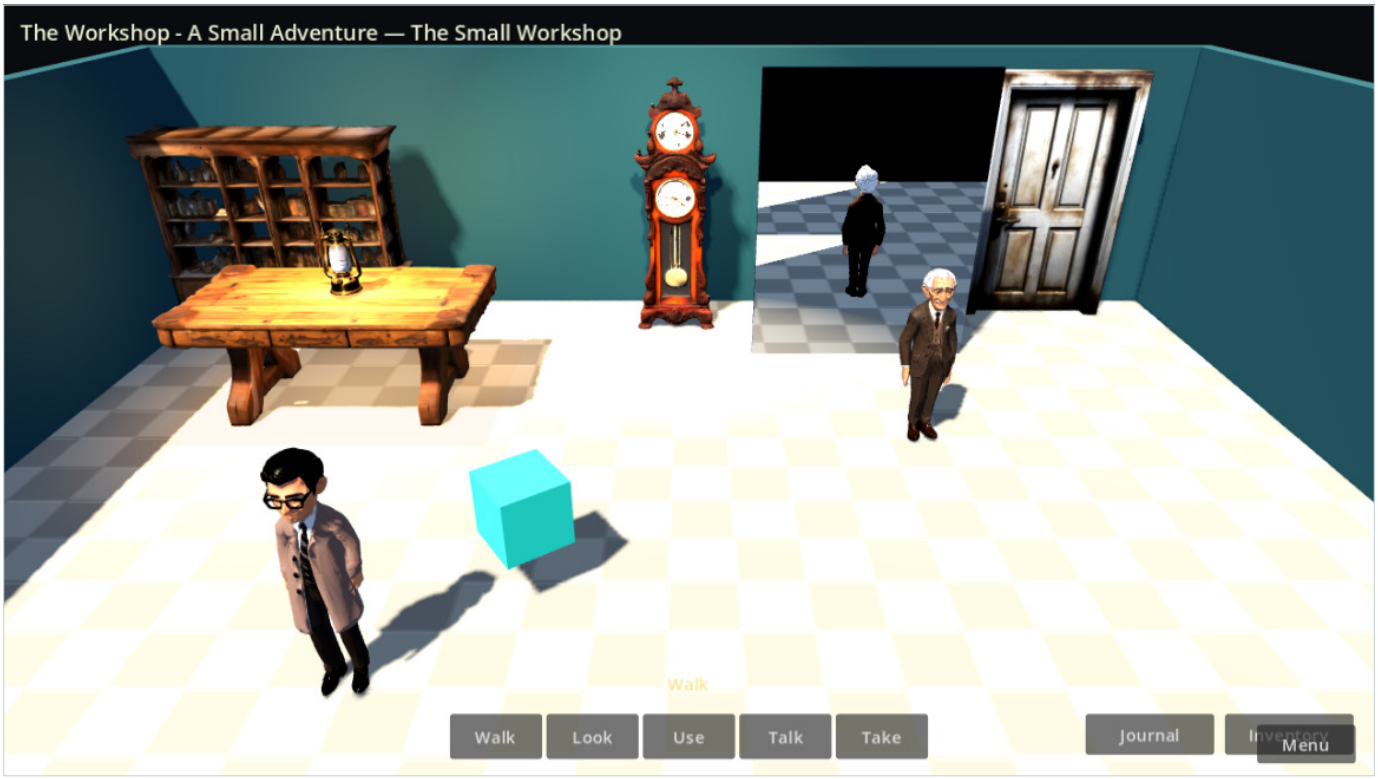


Figure 10. The Group camera frames both workshop characters in a running Test Scene.

Target Offset XYZ lifts or shifts the point being followed. An offset around Y = 1.3 aims nearer a standing character's upper body than the feet. **Follow Damping** is measured in seconds: a larger delay feels softer but less immediate. **Look At Target** can aim a 3D shot at a different subject from the one being followed.

Framed uses screen proportions

The Framed behavior's width and height run from 0 to 1. They describe a zone in the picture, unlike Dynamic Follow's world-unit dead zone. Start around width 0.4 and height 0.3, then walk across the view and watch when the camera begins moving.

A camera on a path

Choose **Path** as the behavior and select a follow target. Scroll to **Camera Path Points**. Click **Add Path Point** for each position you need, then enter its X, Y, and Z values. Use at least two points. For a simple workshop route, try these three:

Point	X	Y	Z
1	-3	3	5
2	0	3	4
3	3	3	5

The up arrow moves a point earlier in the route; the × removes it. Apply after editing. For scripts, the same positions are stored as an XYZ array, such as `[[-3,3,5],[0,3,4],[3,3,5]]`.

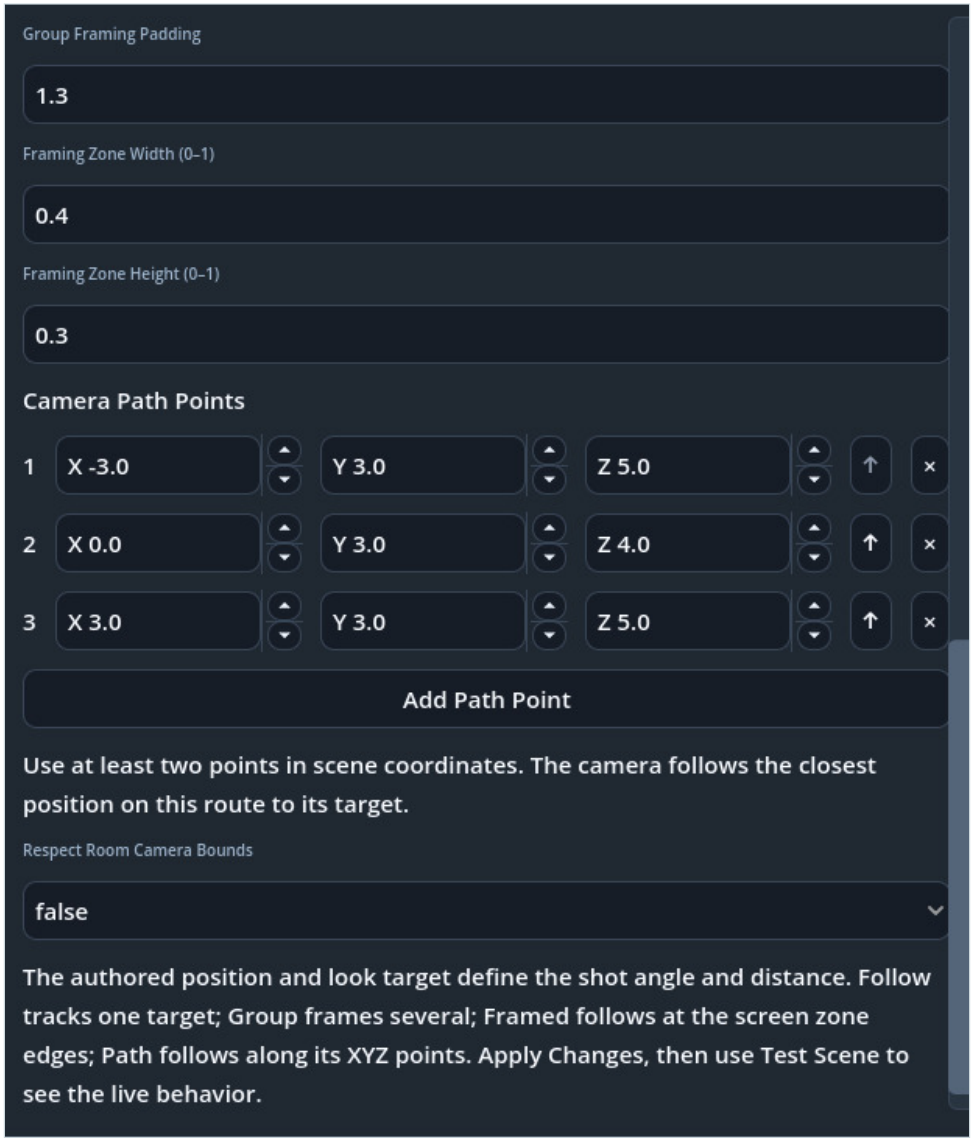


Figure 11. Each row is one camera path point. The up arrow reorders it and the × removes it.

This is a following camera constrained to a route. The points are not a timed shot list. For a timed reveal, activate several cameras in sequence or use the cutscene tools. Do not expect the camera to move to point two after a fixed number of seconds just because it is second in the array.

Blend and choose shots during play

Use **Activate / Blend to Camera** in a verb action, room event, action list, or visual script. Select the camera, enter the blend time, and choose whether the next action waits for the blend. A duration of −1 uses that camera's default; zero makes a cut. **Linear**, **Sine**, **Cubic**, and **Quint** change the feel of the blend.

Set Advanced Camera Behavior changes the active behavior settings. **Set Camera Targets** changes the subjects. **Set Camera Priority** selects among enabled advanced cameras using their priority; equal priorities keep the current shot. An explicit Activate action is easier to reason about for a single conversation.

Enable **Respect Room Camera Bounds** when the shot should obey Chapter 4's limits. A very tight room limit can prevent a group shot from finding its ideal framing, so test those two settings together.

> **Before you continue**
> The shot should follow the intended character, preserve a readable view, and settle when the character stops. Test missing or separated group targets before relying on the shot in a conversation.

7. Focus the shot with depth of field

Depth of field blurs objects nearer or farther than the chosen focus area. It is useful for a close inspection or conversation shot. It is usually a poor place to begin fixing a dark, badly framed scene.

1. Select the camera that will actually be active.
2. Open **Depth of Field** and enable it.
3. Start with a small **Blur Amount**.
4. Enable far blur and place its starting distance beyond the important subject.
5. Adjust **Far Transition** so the blur builds gradually. Add near blur only if the shot needs it.
6. Apply and inspect a running preview.

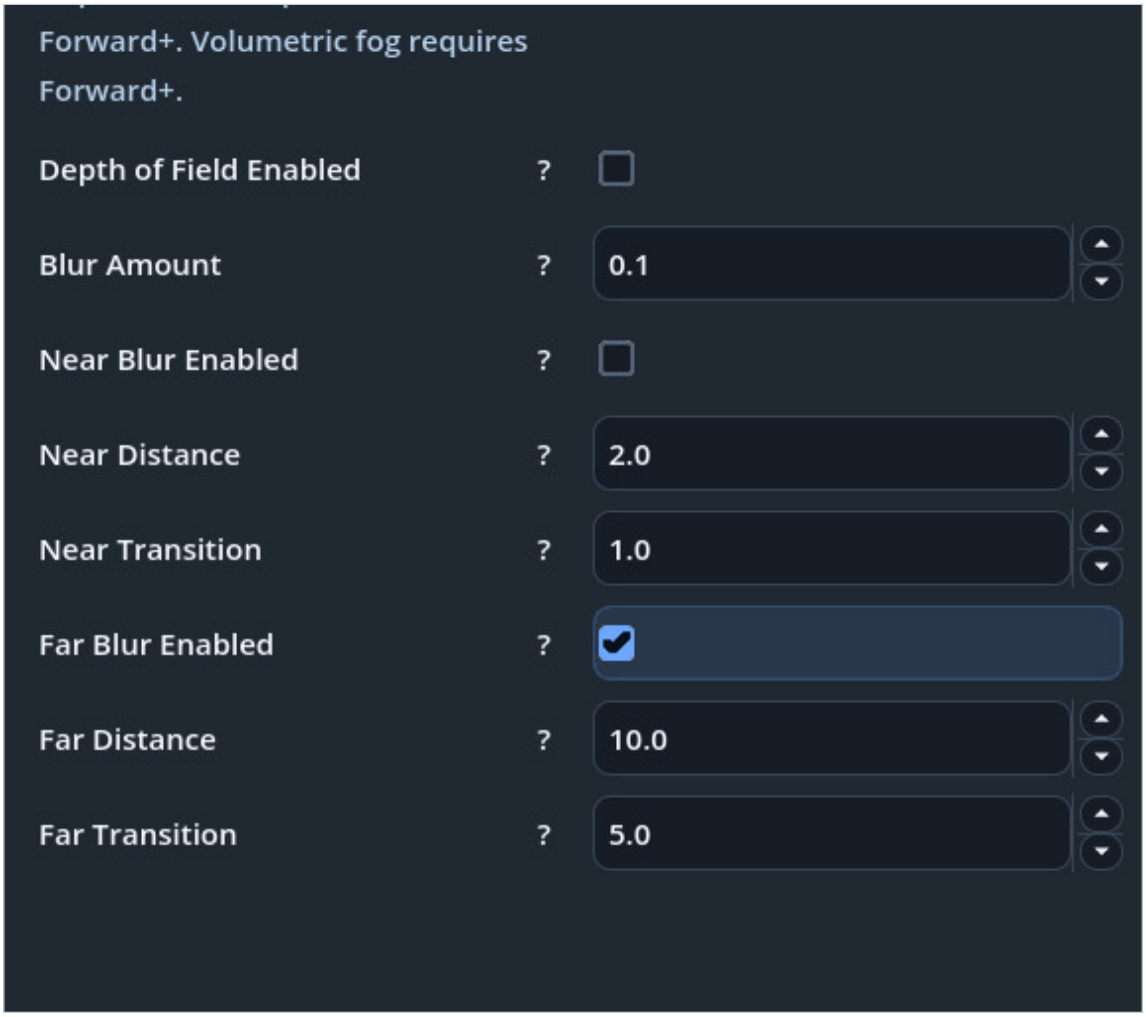


Figure 12. Near and far blur are separate controls on each camera.

Distances are measured from the camera. A character standing at world Z = 0 is not necessarily zero units away from a camera at Z = 14. Use the actual shot when deciding the focus distances.

The room's **Camera & View** page also exposes depth-of-field controls for its default camera. If you switch to a close camera with an action, check that camera's own settings.

In this version, native depth of field needs **Mobile** or **Forward+** rendering. It is unavailable in Compatibility, and also in the transparent layered views identified by the rendering diagnostic. Changing the game renderer does not change an already running editor process. Save, choose the next-launch renderer when appropriate, close PCA, and reopen it. **Play / Restart** restarts the game preview, not the editor's rendering backend.

The screenshots in this guide use Compatibility rendering. They show the depth-of-field controls, not a simulated blur result.

8. Give the room a believable light source

Part 2 added a point light. The other main lights are **Directional** and **Spot**. A directional light acts like distant sunlight. A point light radiates around its position. A spotlight throws light forward in a cone.

Aim a spotlight

1. Choose **Add to Scene > Spot Light**.
2. Give it a useful name, such as Clock Spotlight.
3. Position it above and in front of the clock.
4. Use **Visual Adjustments** to rotate it toward the clock.
5. In **Light Settings**, choose a warm color, a modest intensity, and a range that reaches the subject.
6. Enable **Cast Shadows** if the shot needs them, then test from the game camera.

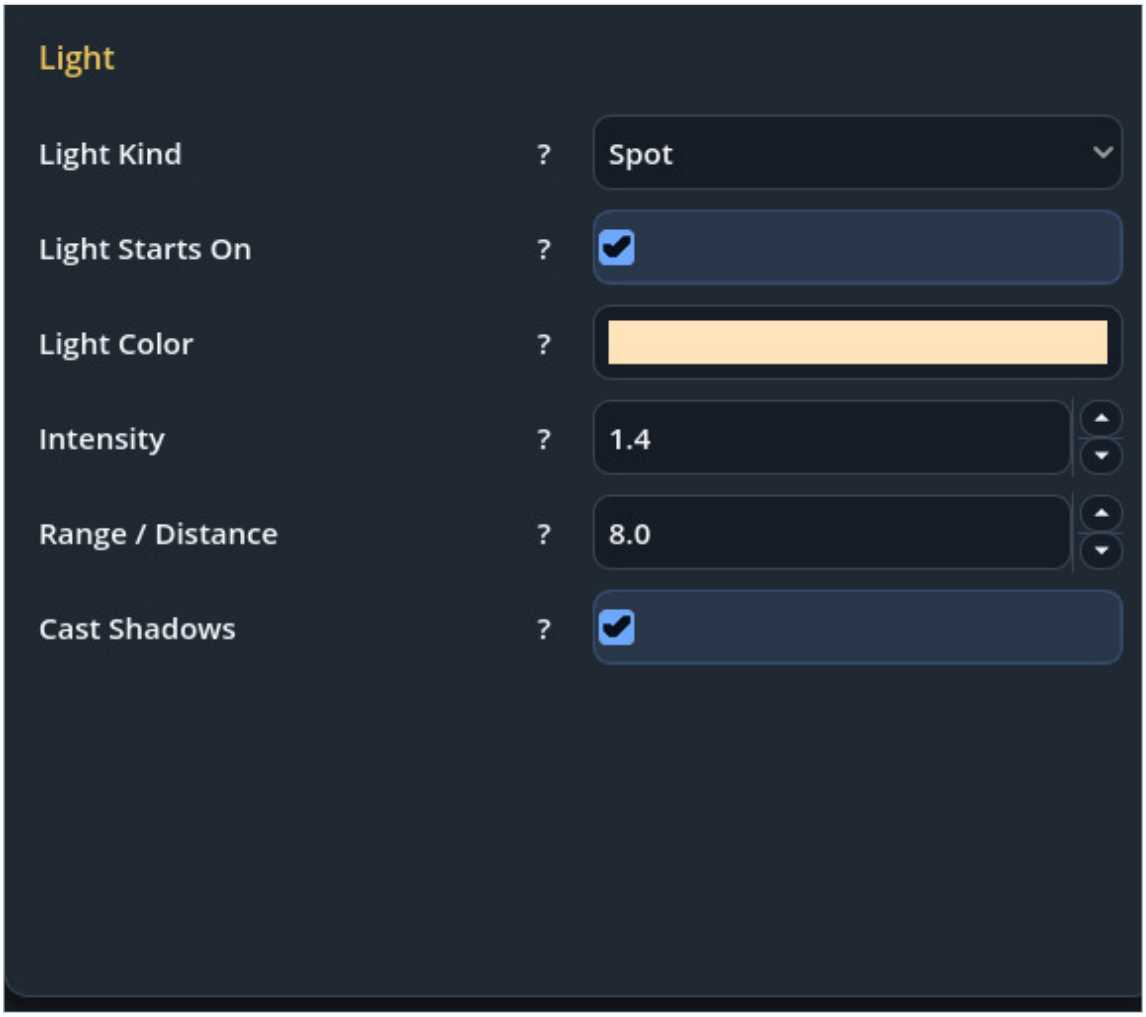


Figure 13. A spotlight has its own position, direction, range, color, and shadow setting.

For this workshop, position near `[0,4,-1]`, X rotation about −0.8 radians, intensity around 1.4, and range around 8 give a starting point. The correct angle depends on where your clock stands. Move and aim the light before increasing its brightness.

The Light Kind selector also contains **Hemisphere**. In this build it provides a broad local fill-light treatment. Do not use it expecting a fully separate sky-and-ground lighting editor.

Brightness and sky

Select the room and open **Lighting & Fog**. **Room Brightness** changes the room-level brightness. Global ambient and rendering settings live in Game Settings; a room can still have its own lights and sky.

Enable **Skybox** when the scene needs a sky. Choose a procedural sky and its sky, horizon, and ground colors, or select a panorama image. **Energy** changes its strength, **Y Rotation** turns the panorama, and **Use Skybox for Ambient Light** lets the sky contribute to the environment lighting.

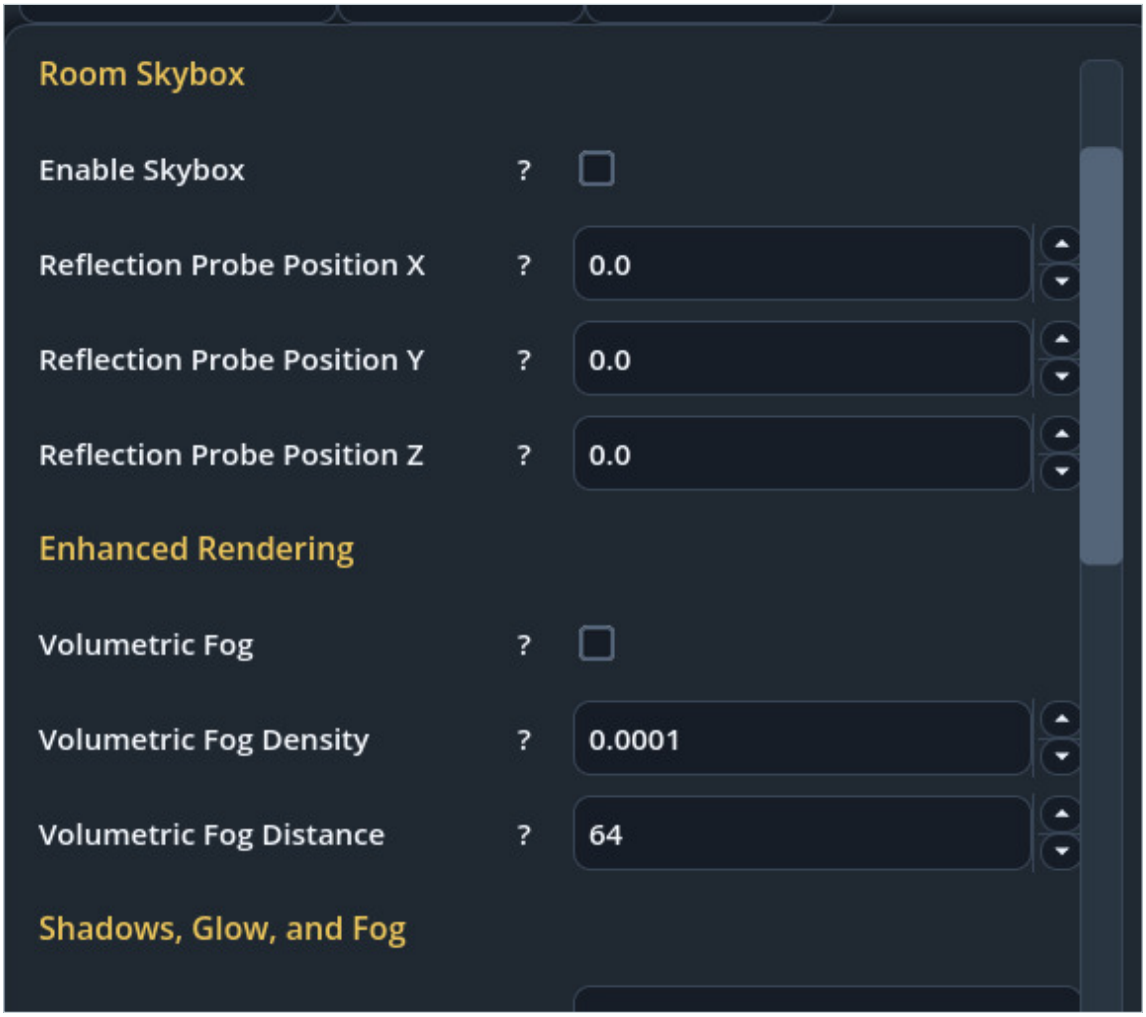


Figure 14. The room can use a procedural sky or an imported panorama.

An indoor room can have a sky configured without the camera ever seeing it. Walls can hide the visible background. That is expected; judge the lighting and the view separately.

Ordinary fog and volumetric fog

For ordinary fog, enable **Fog Enabled**, choose its color, and start with **Depth** mode. Bring **Fog Depth Begin** and **Fog Depth End** close enough to matter in the room. For a small workshop, begin around 5 and end around 25 before refining the result. A fog that starts 100 units away will not help a 12-unit room much.

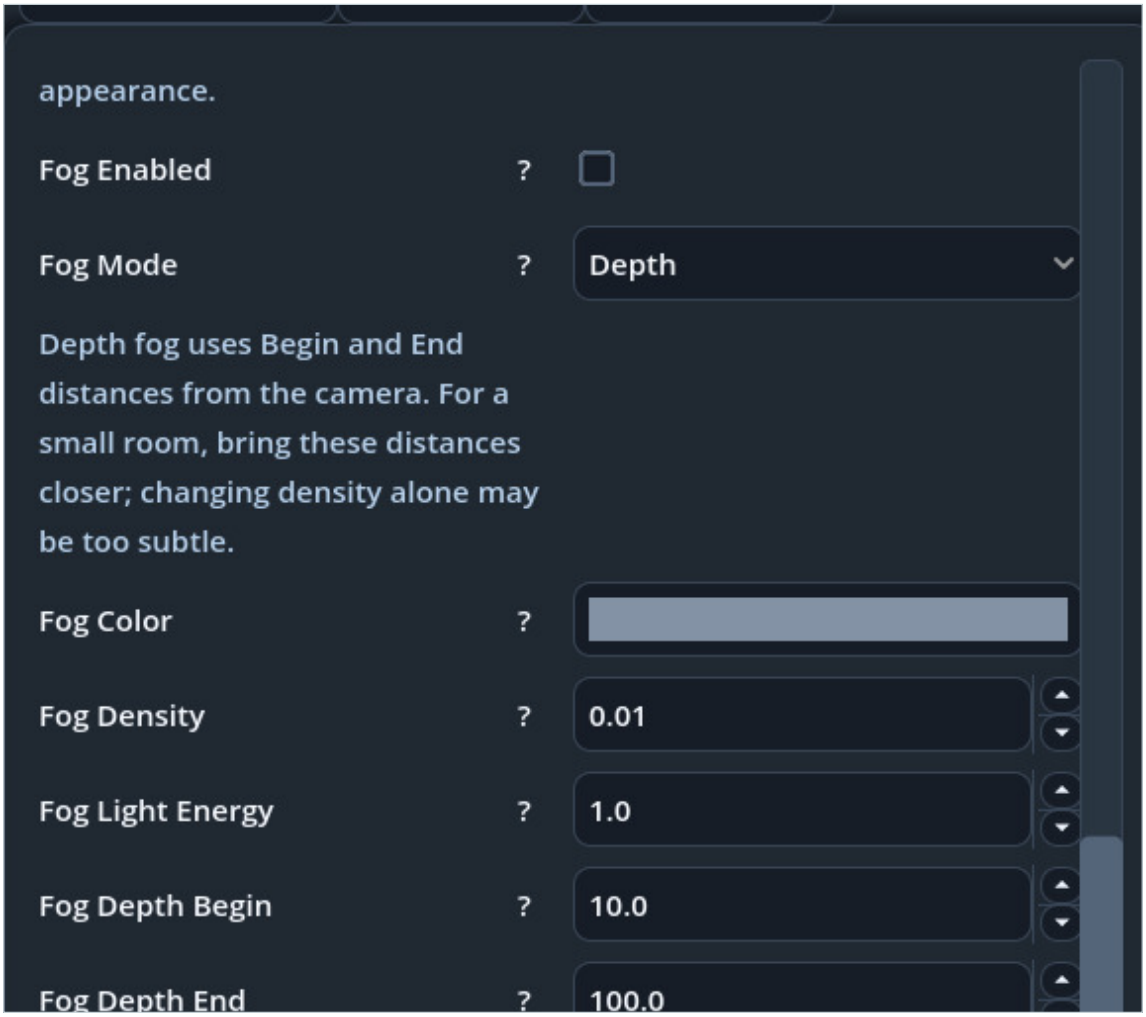


Figure 15. Depth fog is controlled mainly by its camera-distance range; exponential fog uses density.

Exponential mode builds fog with density. Height controls shape fog by elevation. Light energy, sun scatter, sky affect, and aerial perspective refine the atmosphere. Change one of these at a time so you can see what it contributes.

Volumetric Fog creates a different, more demanding effect and needs Forward+. Its density and distance control the volume. Keep ordinary fog usable first, especially when targeting a lower-powered machine.

9. Add flares, shafts, and decorative light strings

Lens flare

Select a light and open **Lens Flare**. Enable it, then begin with low intensity and a small size. The flare is an image-space response to the light; it is not another lamp that illuminates the room.

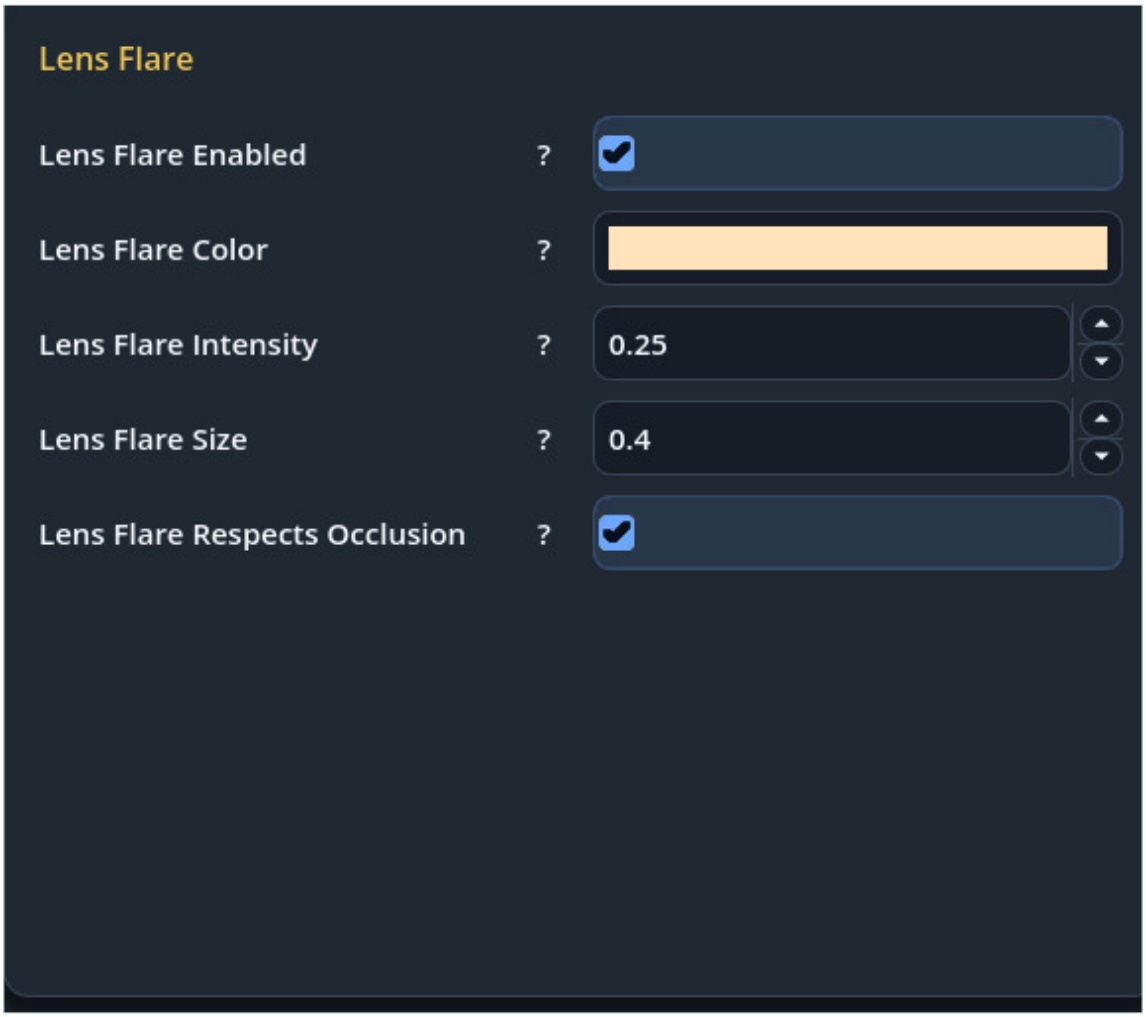


Figure 16. Keep flare size and intensity restrained until the game camera is in the right place.

Leave **Lens Flare Respects Occlusion** enabled when a wall should hide it. If the flare seems missing, check that the light is visible from the active camera and is not hidden behind a solid object.

God rays

The **God Rays** tab controls the light's ray effect: energy, fog density, distance, directionality, and sky affect. It also has **Cinematic Visible Shafts**, with count, width, opacity, and spread controls.

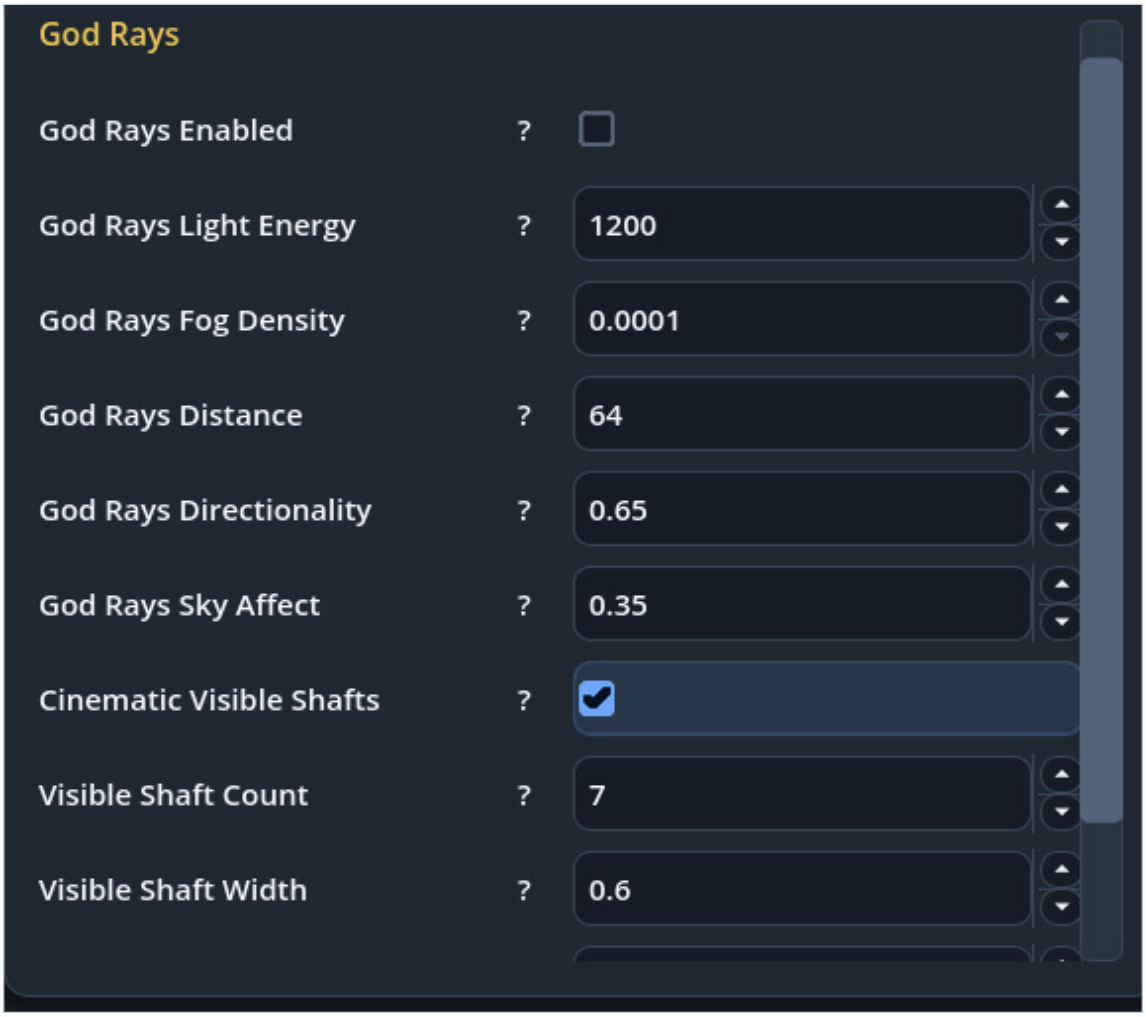


Figure 17. Ray settings belong to the light that supplies them.

Treat the volumetric component and the cinematic visible-shaft controls as separate parts of the effect. Native volumetric fog needs Forward+. A Compatibility screenshot of the settings does not prove that component is being rendered. Start with low shaft opacity, place the light where the view can see the beam, and check the renderer diagnostic before pushing the energy higher.

RGB lighting strings

Use **Add to Scene > RGB Lighting** for bulbs along an axis, or use an ordinary object's **RGB Lighting** tab to add a decorative string to it.

1. Enable **RGB Lighting Enabled** and **Starts On**.
2. Choose the colors, bulb count, spacing, and string axis.
3. Enable **Center Bulbs Around Origin** if the object position should mark the string's centre.

4. Set the bulb size and emission energy.
5. Use **Create A Light Every N Bulbs** sparingly. Zero creates no extra lights.
6. Enable **Cycle Colors** for animation and adjust its speed.



Figure 18. Bulb appearance and actual light emission have separate controls.

A glowing bulb mesh and a light that illuminates nearby objects are different things. The light-every-N option, light energy, and range control the latter. A long string with a real light on every bulb can become expensive quickly.

An optional on/off variable can drive the string. Use `ToggleRGBString` from the action catalog for an interactive switch. Check the action's displayed label in your build and select the actual string record.

10. Make a mirror, not just a shiny object

A bright highlight is not a reflection of the player. **Metallic**, **Roughness**, and **Room Capture** can make a surface glossy, but a mirror needs a reflected view of the moving scene.

Reflection choice	What it is for
Room Capture	General environment reflections on metal or other shiny surfaces. The room probe is configured through rendering settings.
Live Mirror	A planar reflected view that can include moving characters. Use it for a mirror-like surface.
Region Mirror	A mirror surface drawn over a region footprint, useful for a flat reflective patch.

Build a simple wall mirror

1. Add a box primitive and name it Workshop Mirror.
2. Make it wide and tall but thin. Our example uses Size X = 2.3, Y = 3, Z = 0.05.
3. Place it near the wall, with its broad face toward the player and camera. Here its position is `[2,1.5,-2.8]`.
4. Open **Visual Effects** and find **Material and Reflection**.
5. Set **Reflection Mode** to **Live Mirror**.
6. Use **Mirror Axis: Local Z** for this thin-Z box. Automatic chooses the thinnest local axis, which also works for a simple unambiguous box.
7. Start with reflection strength 1, a white reflection color, and low roughness, around 0.02.
8. Apply and test while moving the character in front of it.

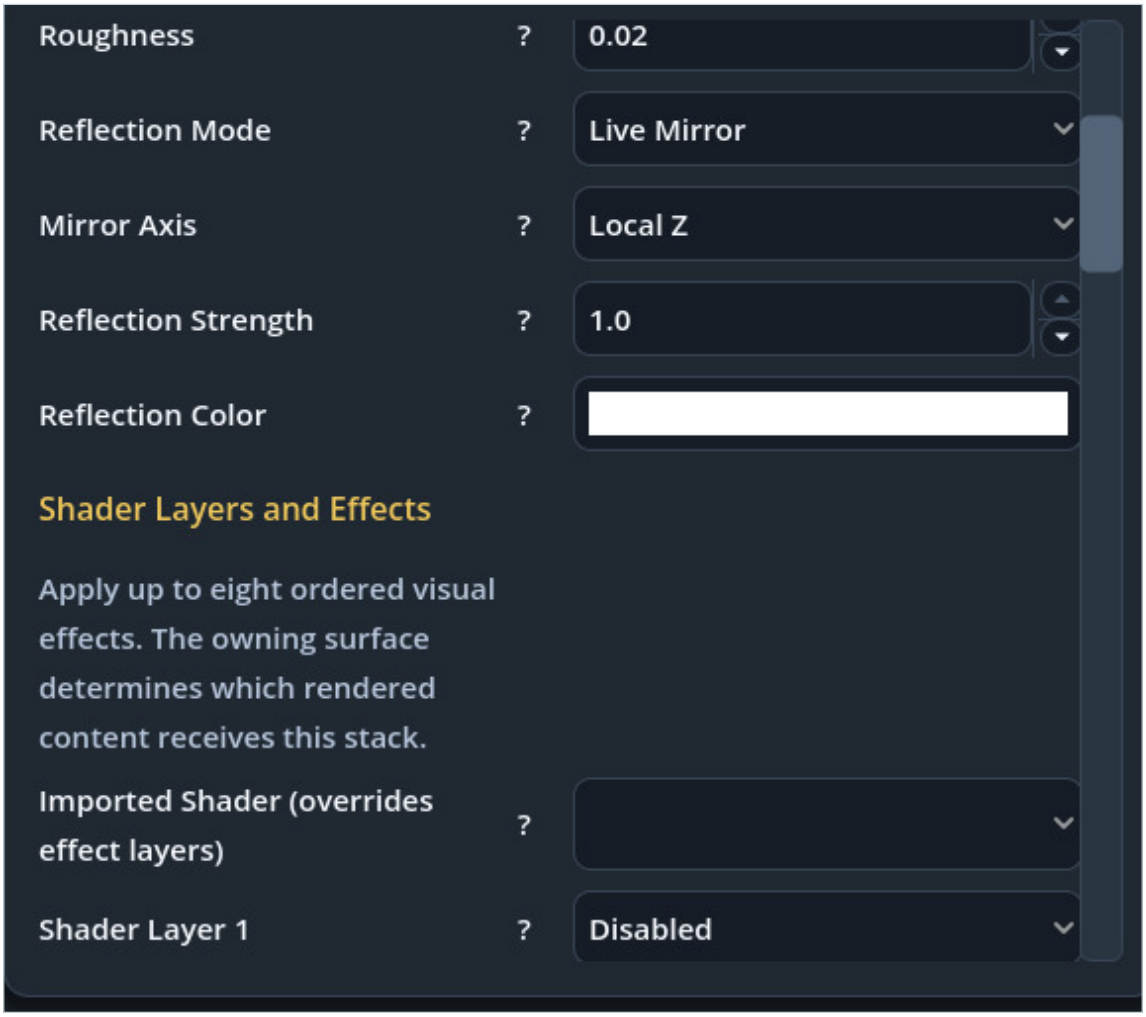


Figure 19. Live Mirror is the setting that distinguishes a mirror from a general shiny material.



Figure 20. The running mirror reflects the backs of the characters standing in front of it, rather than showing only a light highlight.

The axis is local to the object. If you rotate the mirror, the selected local axis rotates with it. It is the direction perpendicular to the broad reflective face, not the direction the character walks along.

Keep the object's opacity and reflection color alpha above zero. Move it slightly forward if another wall occupies the same surface. If it flickers or seems covered, check the overlapping geometry before changing the reflection strength.

For a real imported mirror, a separate, simple reflective panel is often easier to control than applying the effect to the entire decorative frame. A live planar reflection on a rounded statue will not behave like a physically curved mirror.

Room captures and cost

For general glossy materials, enable the room reflection probe in Game Settings' rendering controls. The room's **Lighting & Fog** page exposes **Reflection Probe Position**. Keep the capture inside the relevant space. Low roughness gives sharper reflections; high roughness spreads them out. Metallic also changes the underlying material response.

A room capture is not a guarantee of a constantly updated, correctly aligned reflection of every moving actor. Use Live Mirror for that job. Each visible live mirror needs an additional view, so test performance with the actual number of mirrors you intend to use.

If nothing useful appears, check the mode, reflection strength, tint alpha, mirror axis, object opacity, overlapping wall, and camera angle in that order. Then test a plain thin box before troubleshooting a complicated imported model.

11. Give an object its own surface effect and motion

Room effects change the view. Object effects change a selected object. Use the object's **Visual Effects** tab when only a crystal, a sign, or a machine should look different.

Build a small effect stack

1. Select an ordinary object in Scene Objects.
2. Open **Visual Effects** and scroll to **Shader Layers and Effects**.
3. Leave **Imported Shader** empty while trying built-in layers.
4. Set Shader Layer 1 to an effect, then start with a low intensity.
5. Apply and inspect the object from the game camera.
6. Add a second layer only after the first is useful. Set an unwanted layer to **None** to remove its contribution.

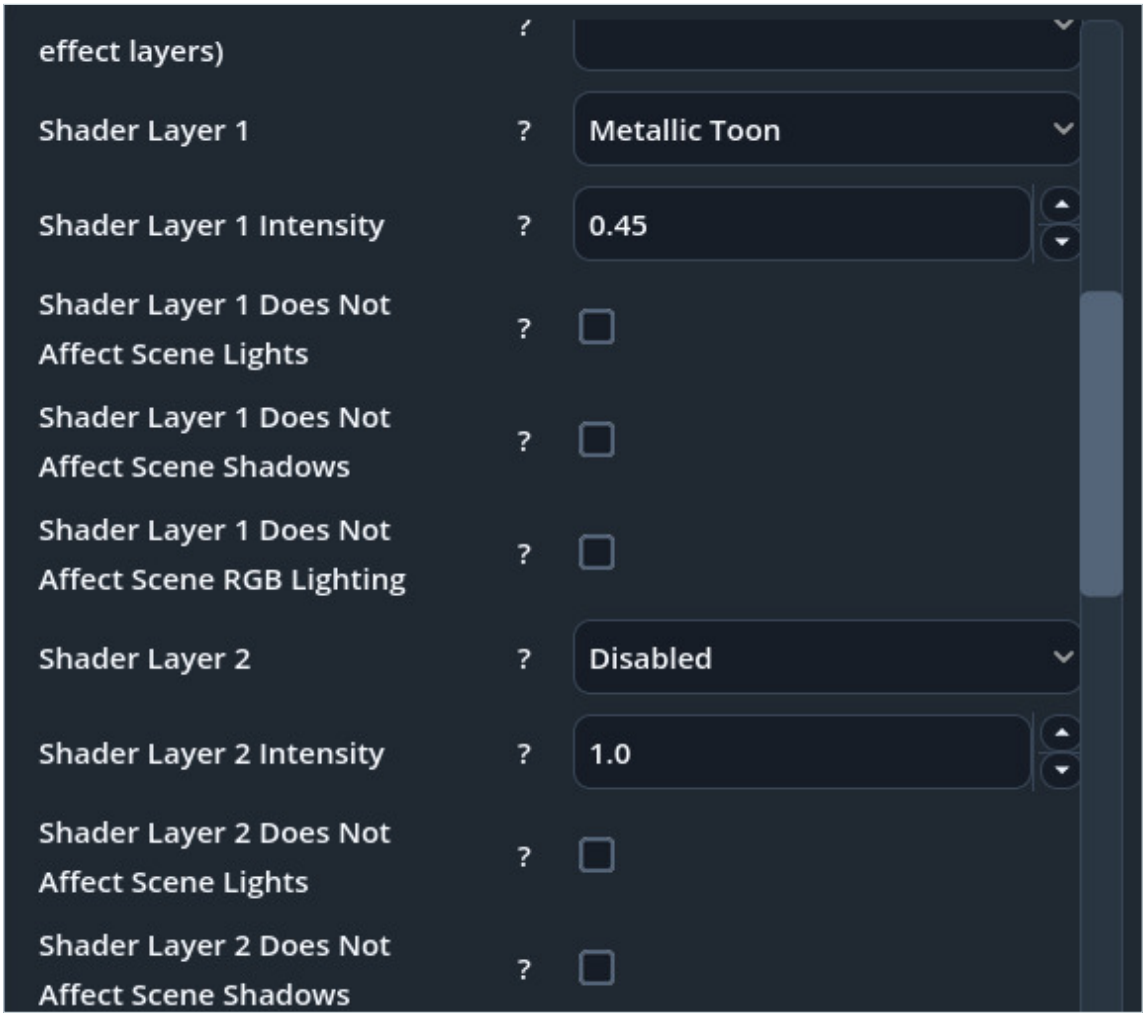


Figure 21. An object can have its own ordered shader layers, independent of the room's screen stack.

There are up to eight layers. Their order matters, so build the look in a deliberate order. A supplied imported shader overrides the built-in layers; do not expect both systems to combine automatically. Object surfaces need a compatible spatial shader. A canvas screen shader belongs on the screen or another supported canvas surface.

The shared fine controls include pixel/grain/stroke size, distortion, animation speed, color levels, edge strength, and tint. They do not all affect every preset. A setting meant for pixelization may do nothing useful to a metallic treatment.

Bloom Lights Surrounding 3D Objects is separate from making the object look bright. Its color, energy, and range control the extra light contribution where that option is supported. Check the actual light on neighboring surfaces rather than judging only the glowing object.

Continuous rotation and pulsing scale

For an ordinary object in this build, find **Continuous Rotation** and **Back and Forth Scale** by scrolling down **General**. These controls are grouped as motion settings but do not have a separate visible Object Motion tab in this version.



Figure 22. Rotation speed is in degrees per second. Scale animation has separate start and end values.

For a rotating display cube, enable **Rotate Continuously**, use Y speed 45, and leave X and Z at zero. **Keep Rotating One Direction** makes a continuous spin. The reverse mode turns through a full revolution and then turns back. A negative speed reverses the direction.

For a gentle pulse, enable **Animate Scale**, set Start Scale to `[1,1,1]`, End Scale to `[1.12,1.12,1.12]`, choose **Grow / Shrink Back and Forth**, and use about 1.5 seconds each way. **Repeat Start-to-End** jumps back to the beginning instead of reversing smoothly.

This is transform animation, not a skeletal character clip. Keep the mesh's normal **Visual Adjustments** scale in mind when choosing motion scales. Use **SetMotion** or **ToggleMotion** from an action list to control it during play, and test the collider as well as the visible movement if the object blocks characters.

12. Help the player recognize useful objects

A good adventure scene needs readable interactions. Start with a clear name and description before adding effects.

Hover attention and cursor changes

1. Select the object and open **Pointer Feedback**.
2. Set **Display Name While Hovering** to inherit the room/global choice, always show, or always hide.
3. Enable **Mousing Over Highlights Object** if it needs extra attention.
4. Choose **Outline**, **Bloom / Glow**, or **RGB Tint**.
5. Set that mode's color and strength. Apply, run Test Scene, and move the pointer over and away from the object.

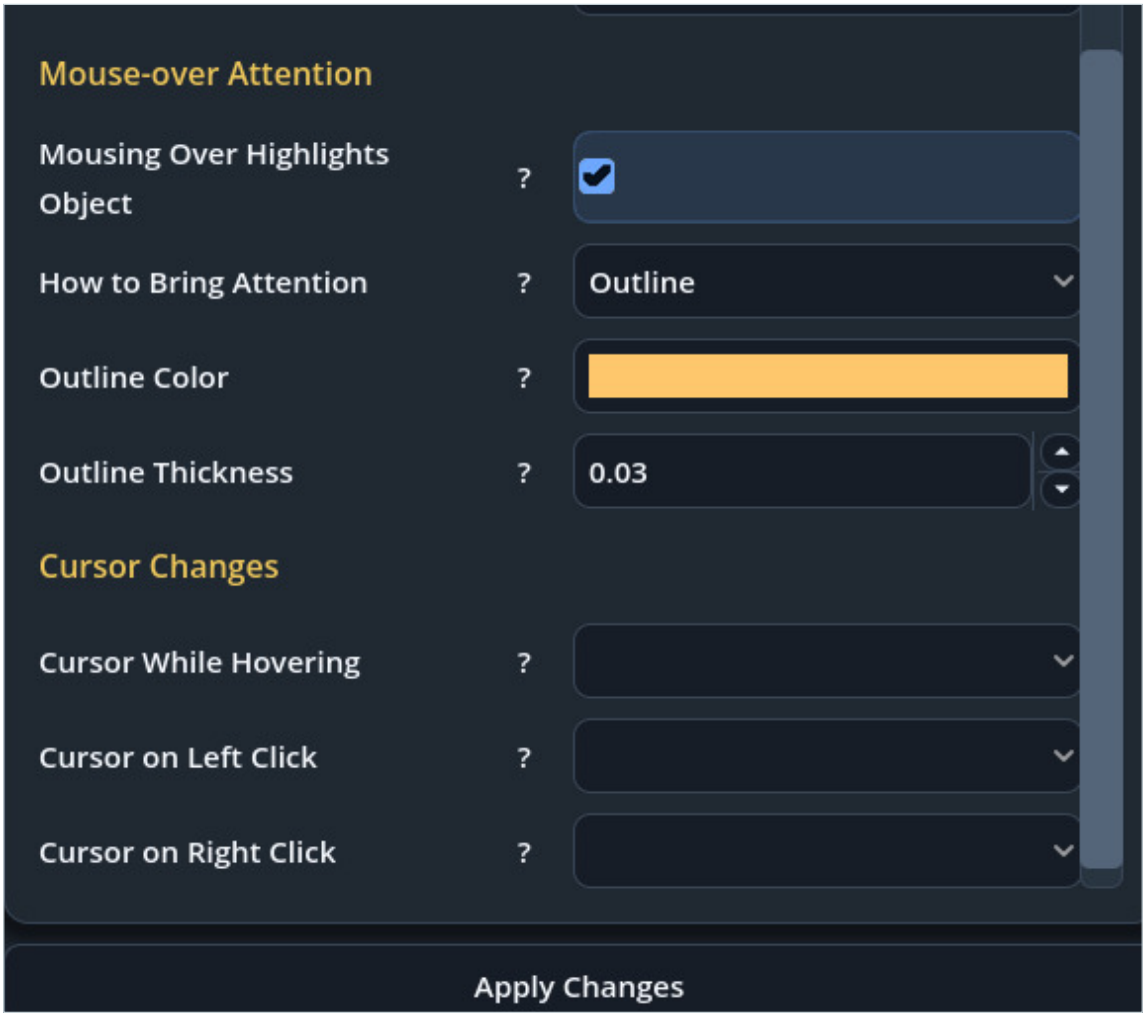


Figure 23. Pointer Feedback controls the response to hovering, including optional cursor changes.

Outline thickness is not a pixel-perfect promise at every camera distance. Judge it in the actual shot. Use a small effect that supports the artwork rather than washing it out.

Farther down this page, **Cursor While Hovering**, **Cursor on Left Click**, and **Cursor on Right Click** can reference cursor records. Create those records in the cursor tools first. These are pointer responses; they do not replace the object's actual verb actions.

Inventory items used on scene objects

The **Inventory Actions** tab handles a different question: “What happens when the player uses this item on this target?” It is separate from the ordinary **Use** verb.

1. Create an inventory definition, such as **Brass Key (brass_key)**, in the inventory tools. Give it a normal image.
2. Make the item available to the player through a pickup or a Give Inventory action. For a quick test, use the project's initial inventory.
3. Select the clock and open **Inventory Actions**.

4. Click **Add Inventory Interaction**.
5. Under **Allowed Inventory Items**, choose the brass key.
6. Set the result action. For example, set the repair variable and show “The brass key winds the clock.” Add the second action with **Add Next Action**.
7. Apply, run the scene, select the inventory key, and use it on the clock.

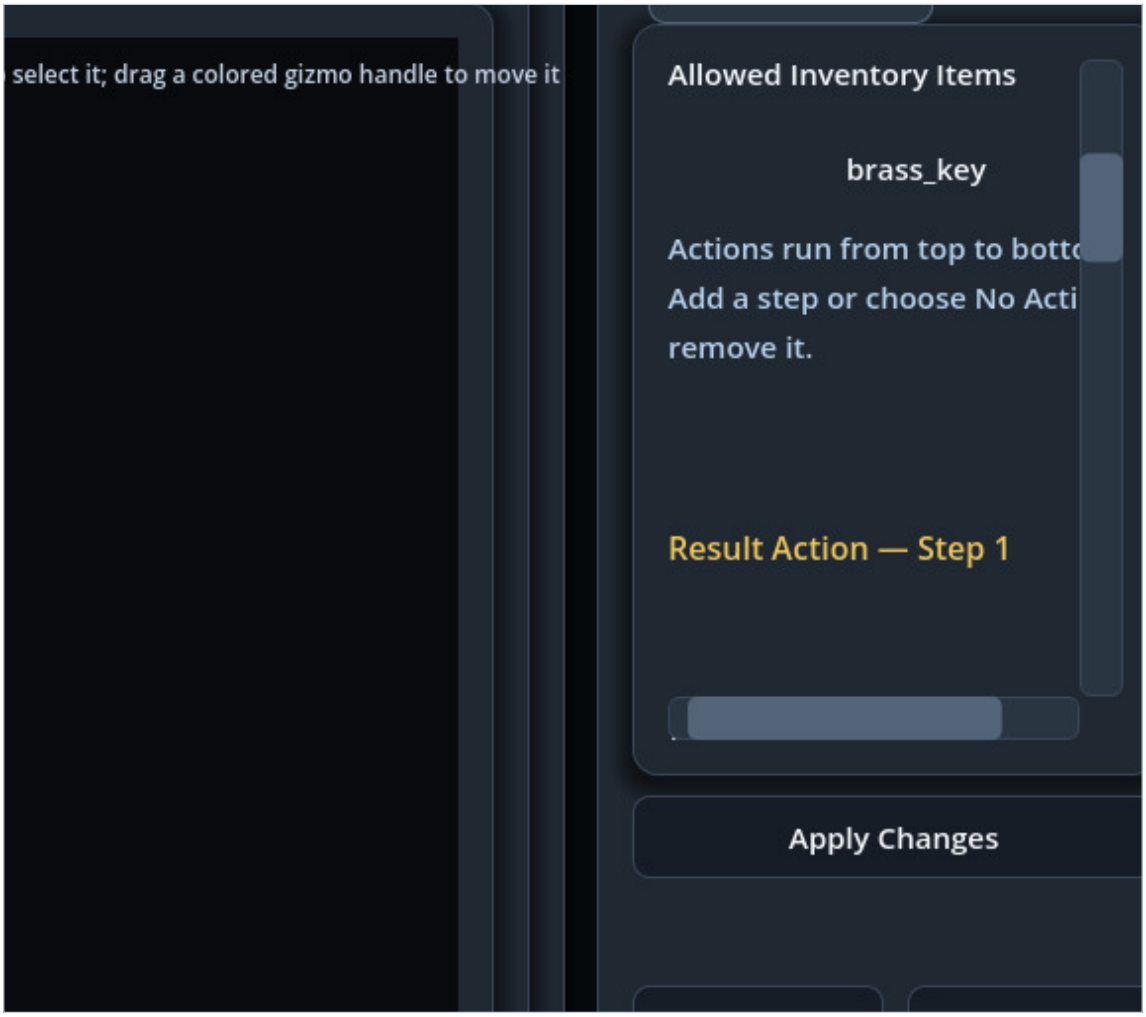


Figure 24. An inventory-on-target rule identifies the allowed item and the result actions.

The selected item and the clicked object both matter. Do not put this rule only in the clock's ordinary Use action and then expect it to reject every other inventory item automatically.

One rule can accept more than one inventory item when those items share the same result. Use separate rules when their results differ. **Remove This Interaction** removes a rule when applied. Whether an item is consumed is an inventory/gameplay decision; using an image as a key does not itself remove the inventory record.

13. Put sound in the room

There are three useful starting points: room-wide audio, a sound attached to a location, and a one-off action sound. Keep those jobs separate so a door click does not accidentally become the room's soundtrack.

Room audio

Select the room and open **General**. Choose **Room Music / Sound**, decide whether to loop it, and set **Room Audio Volume**. This is useful for background music or a broad ambience that should be heard throughout the room.

Use a file that loops cleanly. A pause or click at the end of the recording will be repeated every time the file loops. Test the end of the file, not just its first few seconds.

Proximity audio

1. Choose **Add to Scene > Ambient Sound**.
2. Name it Clock Corner Sound.
3. In General's **Proximity Audio** section, choose a ticking sound asset.
4. Enable **Loop Sound** and **Play in Game**.
5. Start with audible radius around 4 and maximum volume around 0.25.
6. Place it near the clock using **Visual Adjustments**.
7. Test by walking from the far side of the room toward the clock and away again.

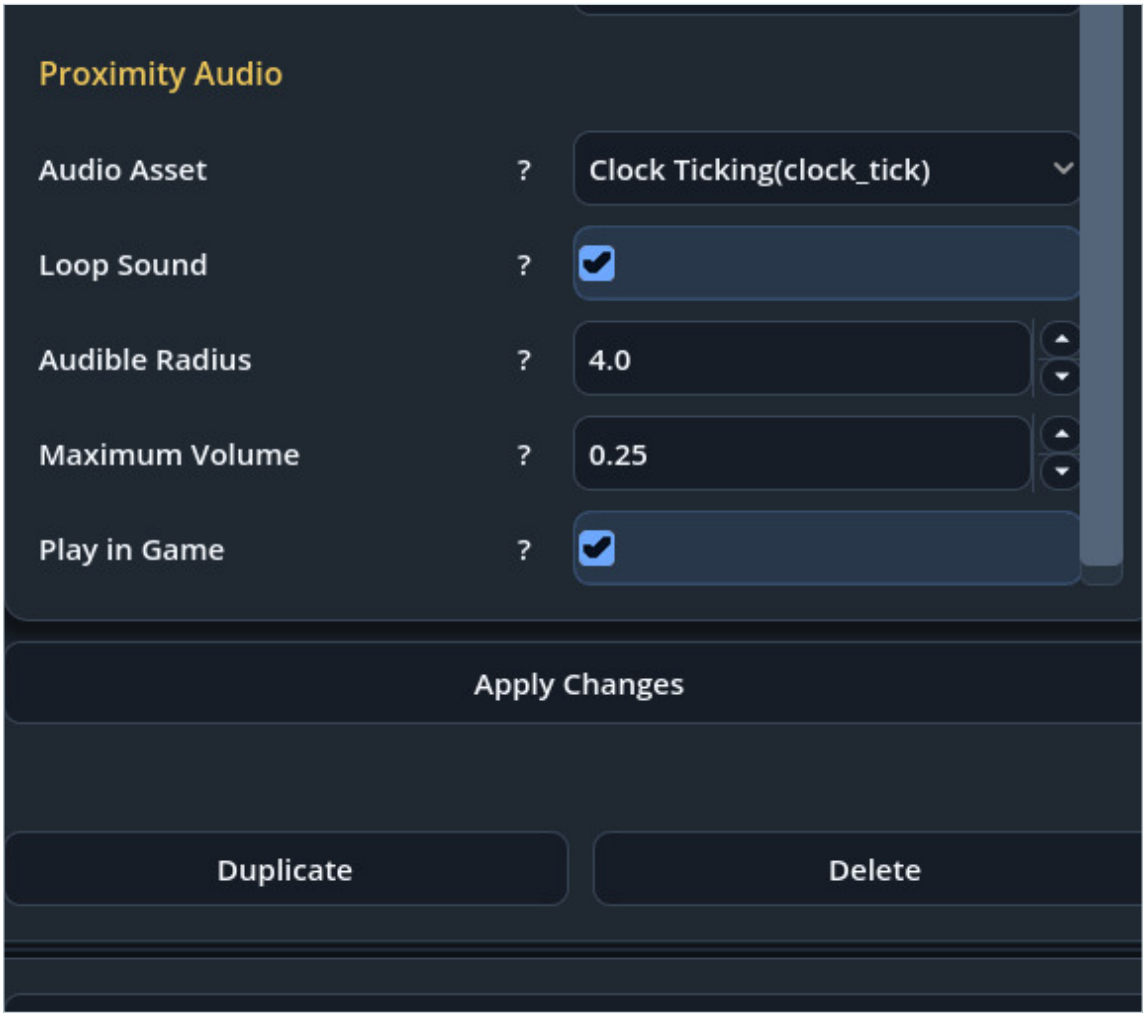


Figure 25. The ambient sound record has a position, an audible radius, and a maximum volume.

The example uses the bundled clock recording. Substitute your own loop if its character suits your scene better. A quiet ticking sound can make a machine feel present without covering speech.

For a one-off chime, use **Play Sound** in the clock's action list. Give it a recognizable channel/ID when you plan to stop, pause, resume, or fade it later. The action catalog also provides music controls. Use the right family when building a sequence so you can control the sound you started.

14. Attach effects and props to moving things

Part 2 placed a flame in the lantern. A placed emitter stays at that position unless you attach it. Attachments let a flame follow a carried lantern, a spark follow a hand, or a prop follow a named point on a character.

Understand an attach point

An attach point is a named position and rotation relative to its owner. A point may optionally follow a bone. A point with no bone follows the object's origin and authored offset; it does not magically find the character's hand.

In this build the placed character/object Inspector appends **Attach Points** near the bottom of **2D / 2.5D**, even for a 3D instance. Scroll to it rather than changing the room's dimension. Reusable asset and character tools also have attachment workflows.

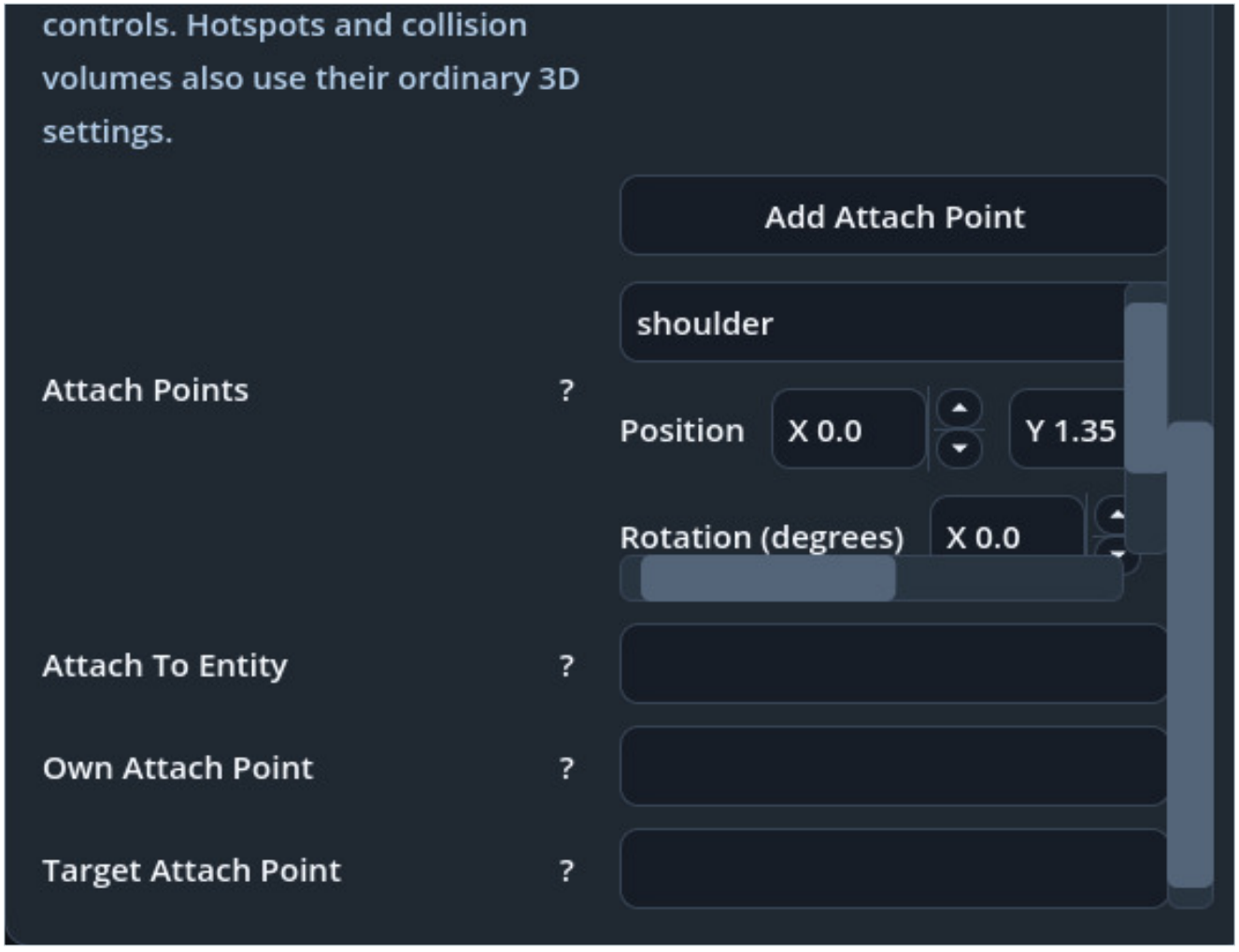


Figure 26. A saved shoulder point has local position and rotation controls.

1. Select the placed player instance.
2. Find **Attach Points** and click **Add Attach Point**.
3. Name the point `shoulder`.
4. Leave Bone Name blank for this first test.
5. Set its position to approximately `[0,1.35,0]`, with zero rotation.
6. Keep the point enabled and apply.

The compact point editor can scroll sideways. Widen the Inspector or use its inner horizontal scrollbar to see all the XYZ values and the bone field.

For a hand-held object, use the actual bone name from the imported skeleton, then adjust the offset against that model. Bone names are case-sensitive references. A name from a different character's skeleton is not a reliable shortcut.

Where the asset/animation preview offers **Place on Object**, it can help place a point visually. The compact Inspector editor may expose only the numeric fields. Animation-specific point overrides are for situations where a shared point needs an authored adjustment in a particular animation.

Make an emitter follow the point

1. Select or add a small particle emitter.
2. Choose its preset before fine-tuning; a preset loads a complete recipe.
3. In **Particle Settings**, choose **Attach To Character / Object: Player**.
4. Enter `shoulder` as **Target Attach Point**.
5. Start with zero **Attachment Offset**.
6. Apply, run the scene, and walk. The source of the effect should move with the point.

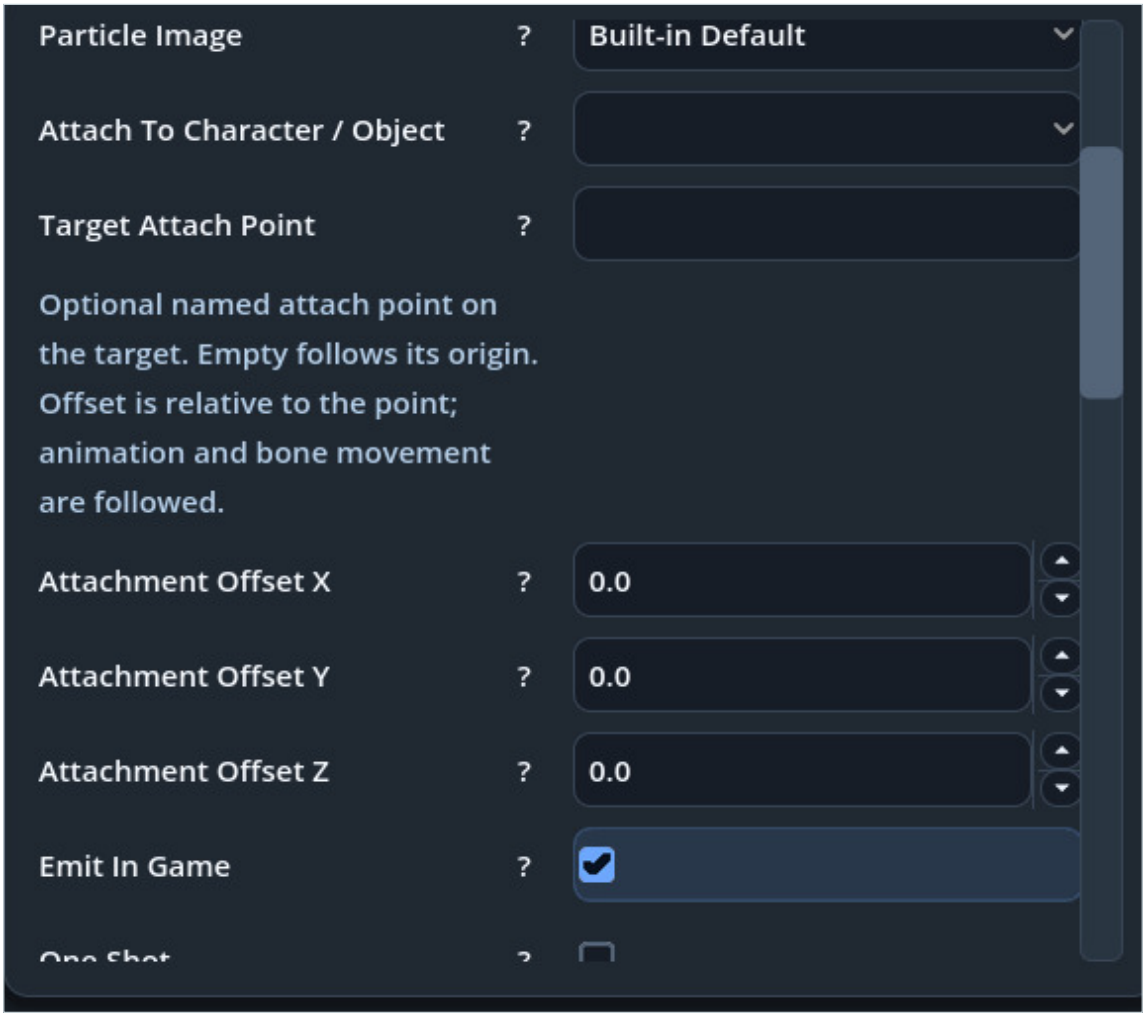


Figure 27. Particle attachment selects a target record, an optional named point, and a local offset.

Use Local Coordinates affects what already emitted particles do when the emitter moves. Try it both ways: local particles travel with the emitter; world-space particles can leave a trail. Neither choice changes which object owns the attachment.

Use **Attach Particle Effect** / **Detach Particle Effect** for a temporary effect controlled by actions. **Attach Object** / **Detach Object**, **Attach Points**, and **Align Attach Points** cover prop and point alignment tasks. Contact actions can compare named points using their contact radii; they do not replace the object's physical collider.

The other particle controls

Controls	What to check
Amount, Lifetime, Size, Opacity	The density, duration, and readability of the effect.
One Shot, Explosiveness, Preprocess	A burst versus a continuous effect, and whether it starts already populated.
Velocity Min / Max, Spread, Direction	The initial movement and its variation.
Gravity, Turbulence	How movement changes over the particle's lifetime.
Emission Area Extents	Where particles begin; this is not the object's event volume.
Particle Image, Draw Style, colors	The visible particle shape and treatment.
Heat Distortion / Heat Motion Speed	Controls used by heat-haze effects.
Bolt Width / Length, flash timing, segments, jaggedness, ribbon	Controls used by lightning-style effects.

Presets include **fire, snow, rain, tornado, wind, hail, heat-haze, lightning, clouds, storm-clouds, bug-cloud, lightning-ball, and explosion**. Use a local emitter for spatial weather or an effect tied to an object. Use a room screen shader for a screen treatment. They are different tools even when both are called snow or rain.

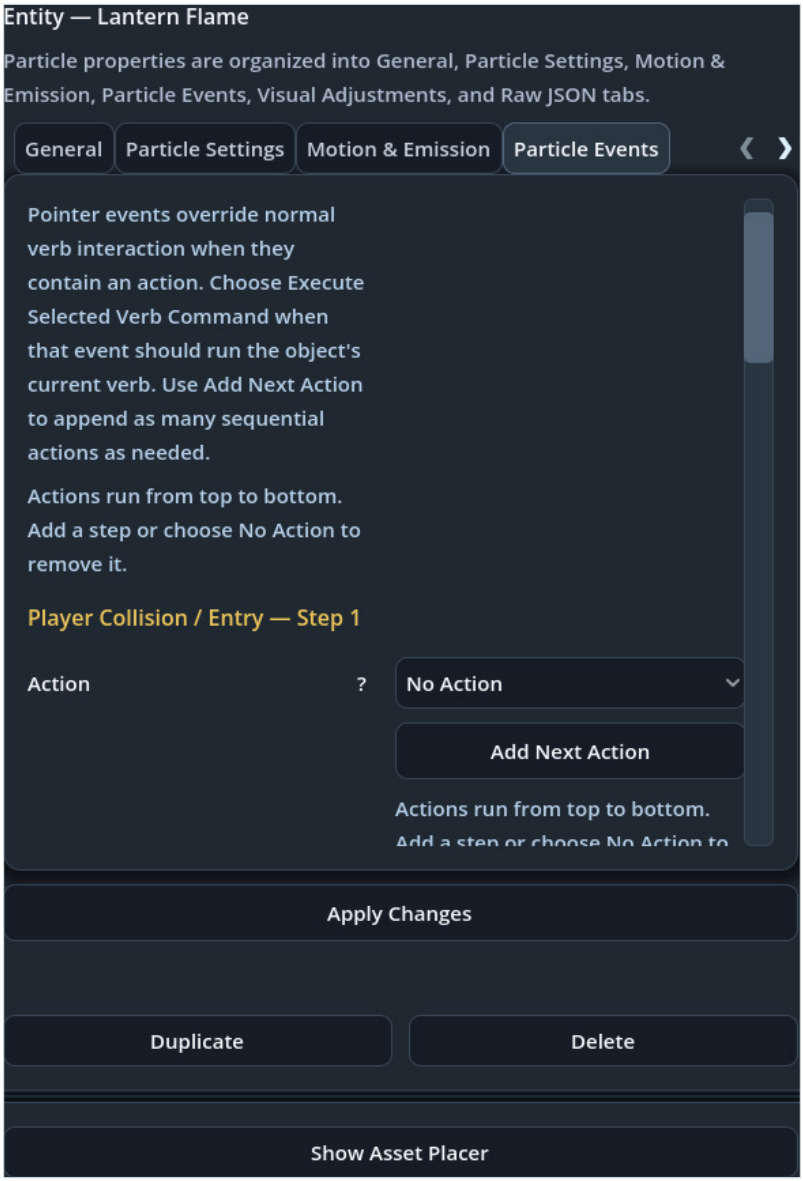


Figure 28. Particle Events uses the familiar entry, exit, pointer, and action-list controls.

Particle Events refers to the emitter object's event volume and interactions. It is not a collision callback for every individual snowflake. Keep its event volume small and intentional if you give the emitter an interactive role.

15. Use regions for local effects and interactions

A region is a defined footprint with an event height and optional appearance. Part 2 used a gentle tint. The same record can supply blur, bloom, a screen preset, a surface material, a mirror, or a custom shader.

1. Add or select a **Region**.
2. Set its footprint and, when events matter, its **Height Above Floor** and **Event Volume Height**.
3. Open **Visual Effects** and enable **Enable Region Effects**.
4. Choose the screen treatment or surface you need.
5. Apply, test the view, then test entry and exit separately.



Figure 29. A region can draw a mirror surface over its footprint. Screen and surface effects are separate sections.

Choice	How to use it
Tint / Blur / Bloom / Preset	Alters the projected region footprint and objects drawn over it. It is not a physical lighting volume.

Choice	How to use it
Custom Screen Shader	Replaces the region's built-in screen effects with a compatible canvas shader.
Surface: Material	Adds a visible surface with color, metallic, roughness, and emission settings.
Surface: Mirror	Adds a planar reflective patch with reflection strength.
Surface: Custom Shader	Uses a compatible spatial shader for that surface.
Surface Height Offset	Lifts the surface slightly above the floor to avoid overlapping it exactly.

A region's blur does not promise to isolate only the pixels of one particular object. The screen footprint can cover objects drawn over it. Use the object's own shader when the effect must belong exclusively to that object.

Region Events supplies Player Collision / Entry, Player Exit, and pointer events. **Verb Actions** supplies the authored verbs. A region is not automatically a solid wall; use blocking or collision for that. Conversely, a Blocked Area is a navigation exclusion and does not provide this effects-and-events editor.

Start with one region and one mild effect. Then cross its boundary, click inside it, and check an object partly overlapping its edge. This reveals how the footprint behaves before you build a puzzle around it.

16. Build a room screen-shader stack

Select the room, open **Post Process**, and scroll to **Room Screen Shader Stack / Post Process**. This is where PSX, N64, and Horror Fog belong when you want them to affect the scene as a whole.

1. Set **Scene Shader Source** to **Use This Scene's Shader Stack**.
2. Leave **Imported Scene Shader** empty while using the built-in layers.
3. Choose **Layer 1 Preset** and start with a moderate intensity.
4. Apply. Turn on **Shader Preview** in the Scene Designer if you want to inspect the effect there.
5. Use Test Scene to check the actual game view and its GUIs.
6. Add another layer only when you can explain what it contributes.

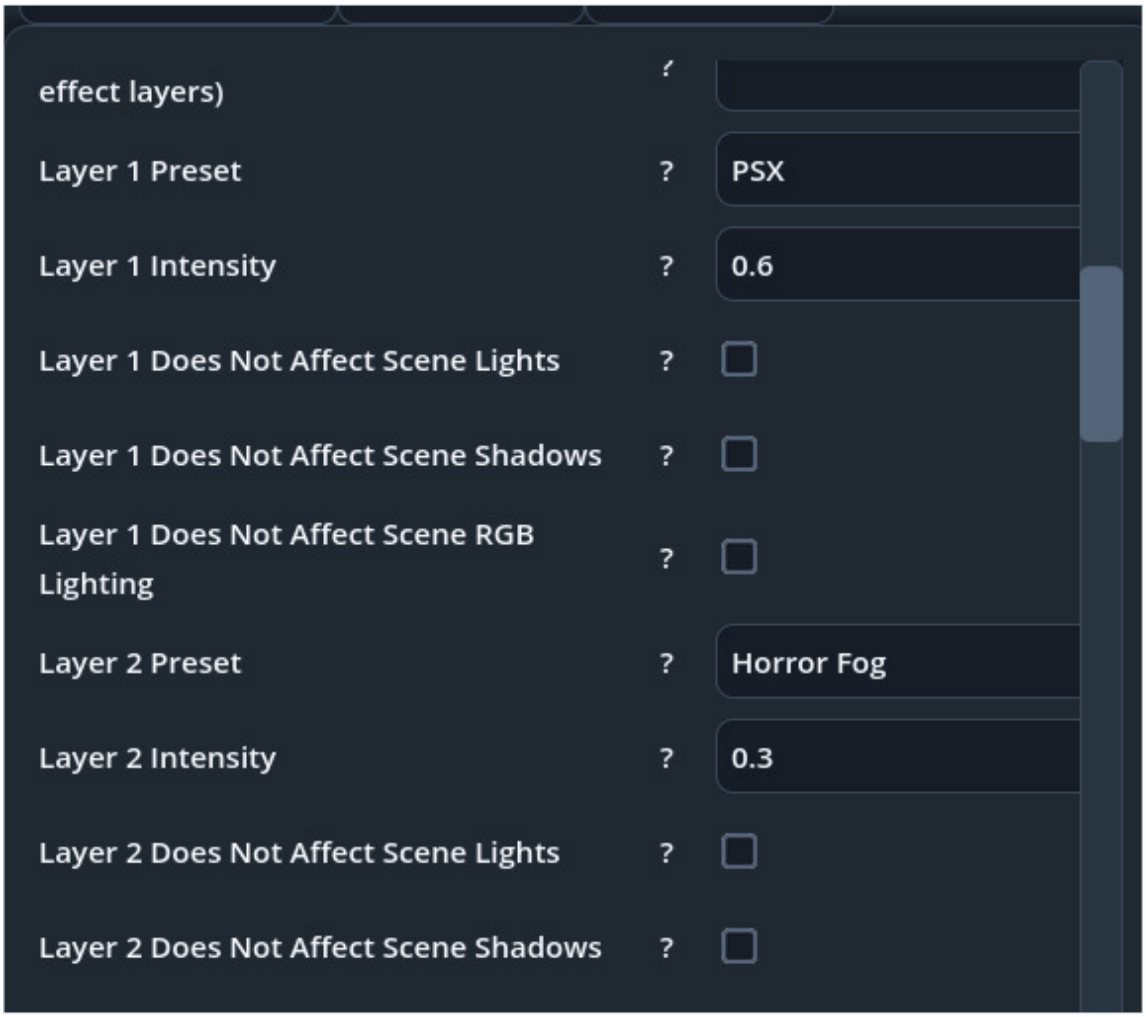


Figure 30. The room owns this screen stack. Imported scene shaders override the built-in effect layers.

Use Global Scene Shaders inherits the project stack. **Disable Scene Shaders in This Scene** disables this room's scene stack. These options let one room have a different look without rewriting every object's material.

Up to eight layers run in order. Each has an intensity and options to exclude scene lights, shadows, or RGB lighting from that layer's treatment. Those options control the effect's scope; they do not turn the underlying lights or shadows off.

PSX, N64, and Horror Fog



Figure 31. Retro and horror presets have their own controls below the layer list.

Preset	Main controls
PSX	Pixel size, color bits per channel, dither strength, and optional vertex jitter.
N64	Pixel size and three-point filter strength.
Horror Fog	Amount, start/end distance, color, film grain, and vignette.

Choose the layer preset first, then adjust the controls for it. More jitter is not automatically more authentic or more playable. Test thin objects, animation, and small distant details before committing to a strong retro treatment.

Horror Fog has camera-distance controls. Bring the end distance inside the scene's useful range before deciding it is missing. A dark fog with a long end distance can be very subtle in a bright, short room.

Other looks and temporary zoom blur

The stack also offers pixelization, distortion, black-and-white, toon and outline treatments, low-poly and art styles, sepia, posterization, vignette, CRT/VHS, film grain, dithering, hologram, bloom, invert, warm/cool, weather, underwater, and stencil drawing. The complete preset index is in Chapter 29.

An imported **canvas** shader can replace the room stack. Import it through Assets, confirm its intended shader type, then choose it in **Imported Scene Shader**. Leave it unassigned if you intend to use the eight built-in layers.

Use **Zoom Blur** from an action list for a brief impact, a sudden discovery, or a transition accent. Set the duration and a modest strength in that action's own fields. Use **Stop Zoom Blur** when a sequence needs to stop the temporary effect explicitly. It is an action-driven effect, not another permanent room-layer preset.

The scene stack affects the rendered room. GUIs and overlays stay outside it unless their own settings explicitly request room shaders. That keeps an old-TV look from automatically making every menu label difficult to read.

17. Weather, outlines, and cinematic grading

These controls also live in the room's **Post Process** page. Work on them after the basic lighting and camera are readable.

Screen weather and surface snow

Choose a **Snow** or **Rain** layer before adjusting its detailed controls. Density changes how much appears; speed and angle control travel; scale or streak length changes the visible shape. Color includes alpha, so a nearly transparent weather color can hide an otherwise enabled effect.

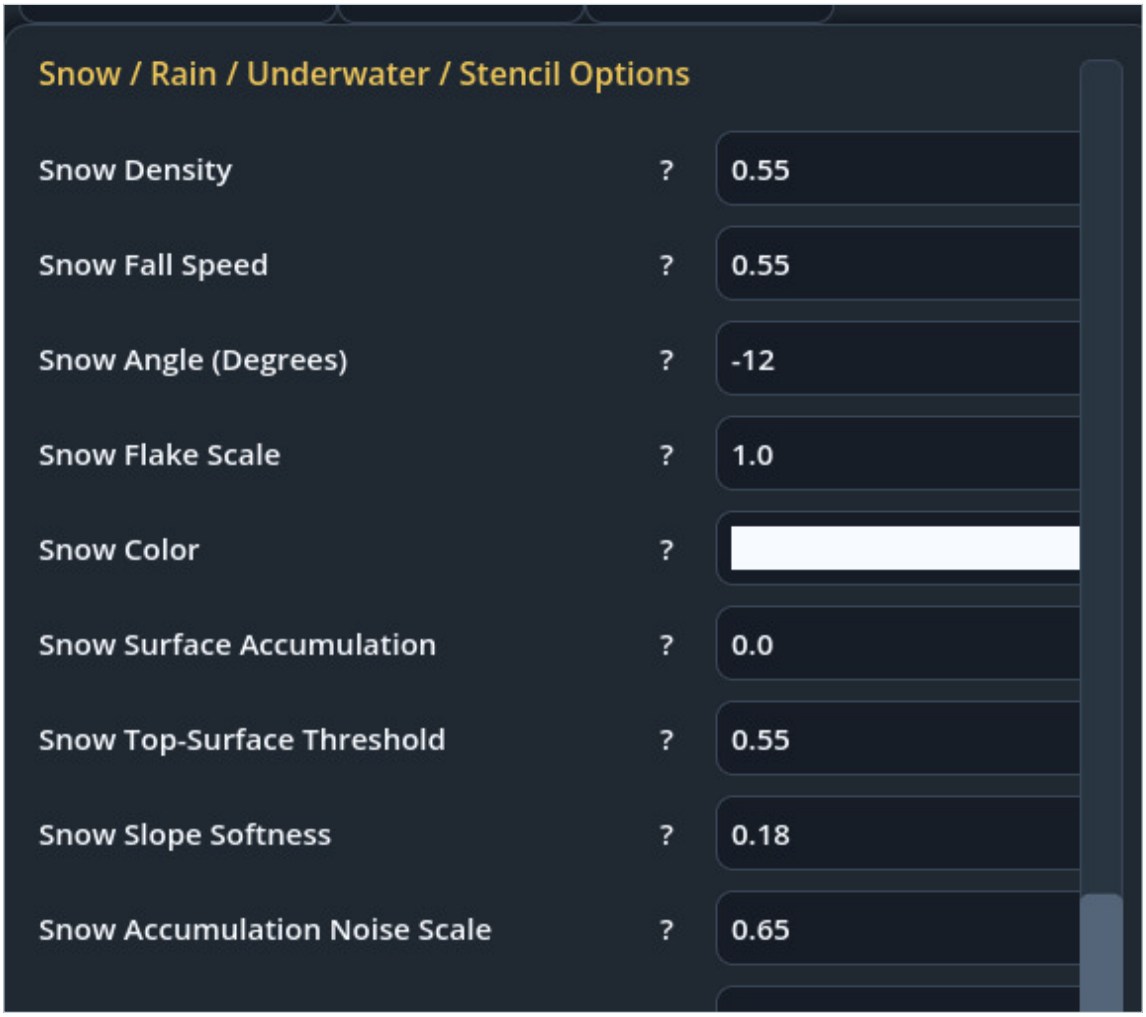


Figure 32. Weather controls include screen movement and a separate set of snow-accumulation settings.

Snow Surface Accumulation is separate from falling flakes. Its slope threshold and softness determine which surfaces receive it. Noise scale and patchiness break up an even coat; accumulation time controls how long it takes to reach the authored amount. The target object's **Receive Snow Accumulation** checkbox allows an object to opt out.

For underwater treatment, tune tint strength, distortion, wave speed, and caustics after selecting the preset. For stencil drawing, use ink/paper colors, darkness threshold, edge strength, and line width. Do not adjust these controls expecting them to alter an unrelated CRT layer.

Whole-scene outline

Enable **Outline Everything in This Room** for the shared outline treatment. Choose its color and thickness, then compare the scene with **Edges Only** off and on.

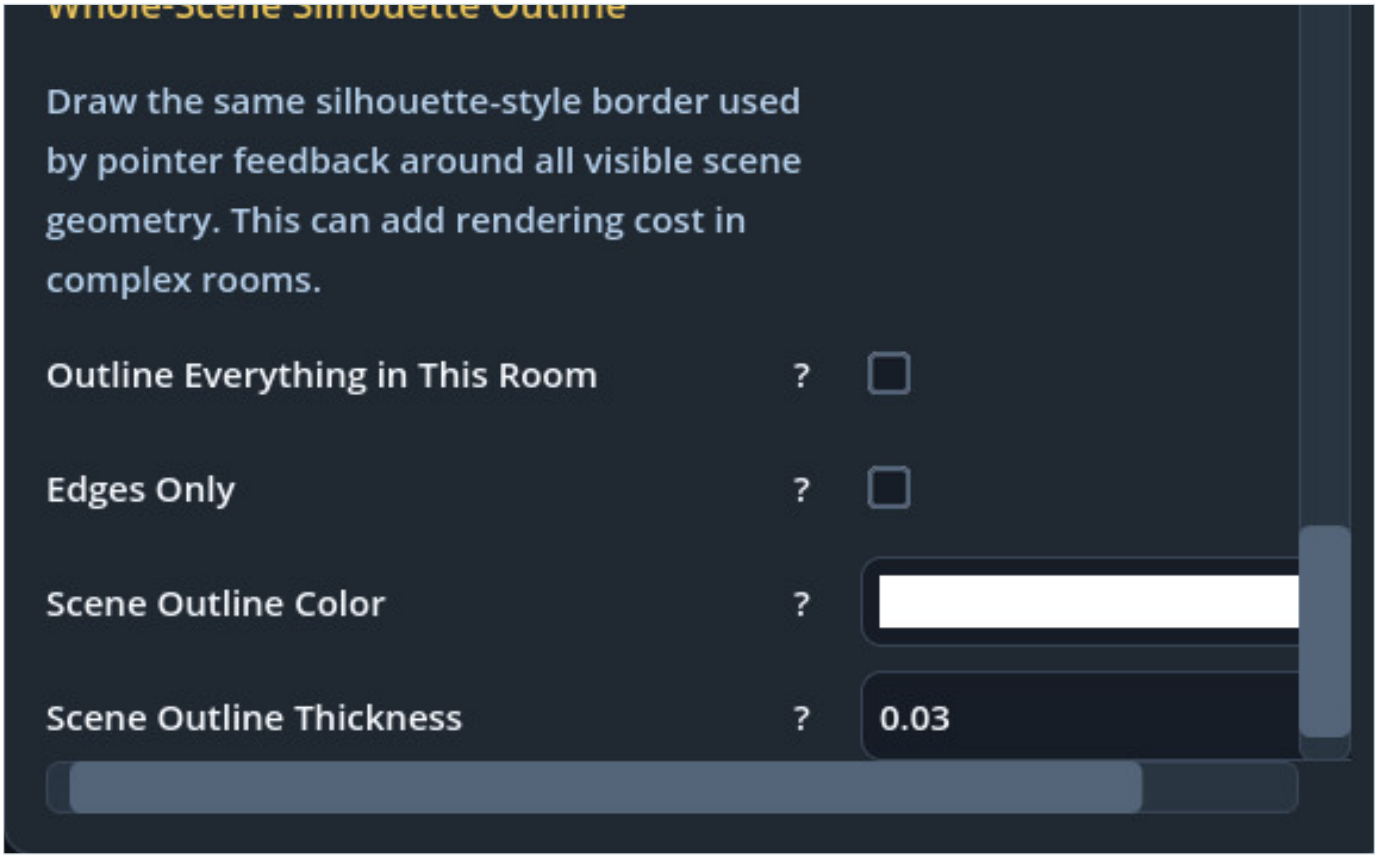


Figure 33. Edges Only changes the whole-scene silhouette treatment without changing an individual object's hover outline.

Use **Edges Only** when you want the boundary treatment rather than a filled silhouette-style surface treatment. Judge it from the game camera, especially on imported meshes with many separate parts. This setting is different from Pointer Feedback, which appears only while a particular object is being pointed at.

Cinematic presentation

At the top of Post Process, set **Cinematic Override** to Custom and enable **Cinematic Presentation**. The controls are exposure, contrast, saturation, warmth, vignette, film grain, and letterbox height at each edge.

Exposure brightens or darkens the grading; it does not move a lamp. Saturation removes or strengthens color. Warmth shifts the color feel. Letterbox covers world edges rather than the GUI. Use small changes and keep hotspots legible. If the player has to squint to find a door, the atmosphere has started charging admission.



Figure 34. Cinematic presentation supplies grading and letterbox controls separately from the room's shader layers.

A room can inherit the project grade, supply its own, or disable it. Save a plain version before combining strong grading, weather, retro layers, and outlines. Otherwise it becomes difficult to tell which control is responsible for a problem.

18. Add a screen overlay and choose the room's view size

A screen overlay is useful for a foreground frame, a vignette image, or decorative artwork that belongs over the view. It is different from a walk-behind plane inside the scene.

A room-specific overlay

1. Import a suitable image. Use transparency for parts that should leave the scene visible.
2. Select the room and open **Screen Overlay**.
3. Set **Room Overlay Mode** to **Use Custom Overlay in This Room**.
4. Choose **Room Overlay Image**.
5. Choose **No Shaders**, **Use This Room's Shaders**, or **Use Overlay's Own Shaders**.
6. Decide whether screen transitions should affect the overlay.
7. Apply and test the active camera, a camera move, and a room transition.

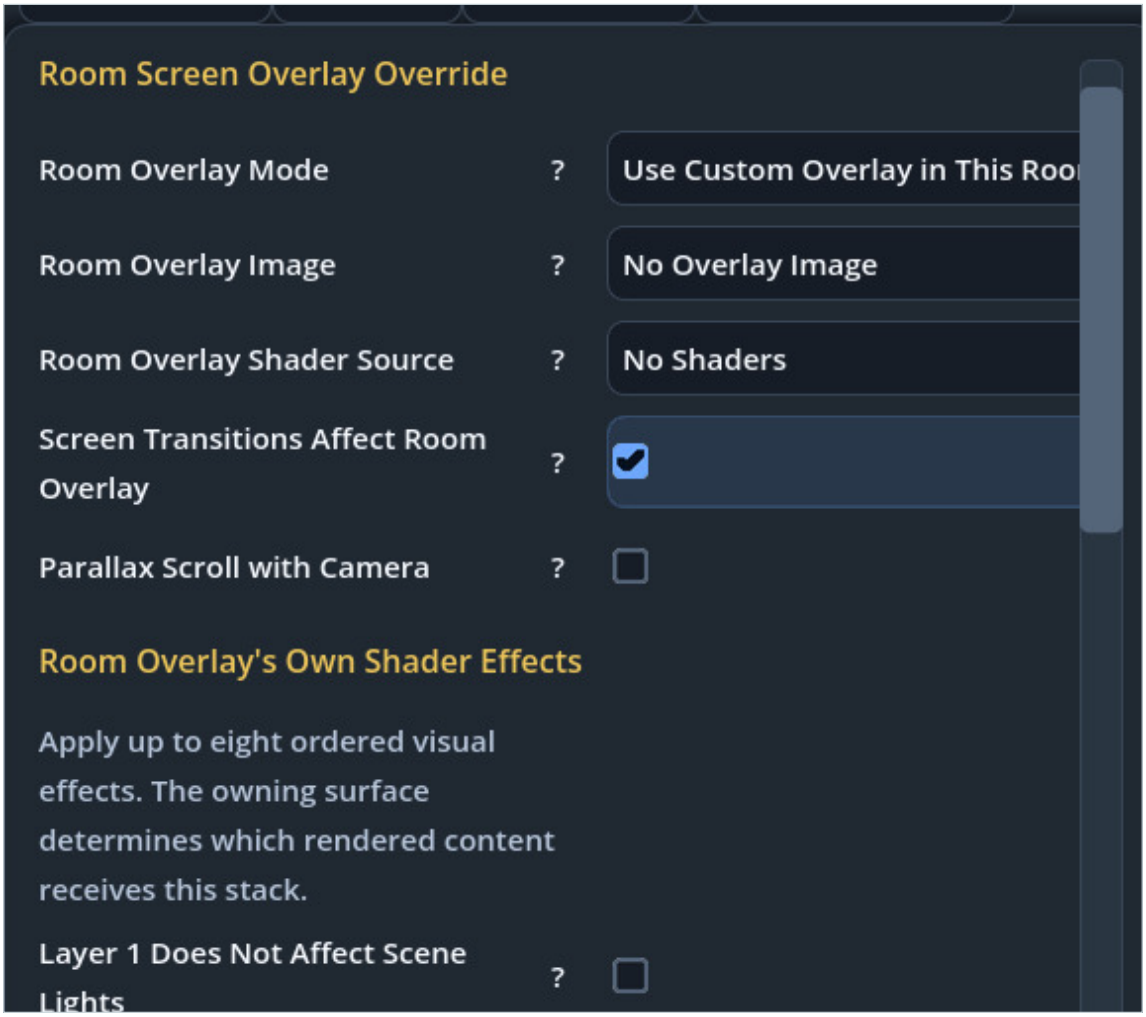


Figure 35. Room overlays can inherit, be disabled, or have their own image and shader treatment.

Enable **Parallax Scroll with Camera** if the overlay should shift as the camera moves. Choose the axes, then start with a low travel strength and comfortable smoothing. The effect needs camera movement to reveal itself; a still screenshot cannot demonstrate it.

The overlay's own shader source exposes another ordered stack. This changes that overlay's appearance, not the entire room. Keep decorative art clear of important buttons and text.

Room viewport size

Select **Camera & View** and find **Room Viewport Override**. Most rooms should use the global project size. Choose a custom width and height only when the room needs its own logical render dimensions.

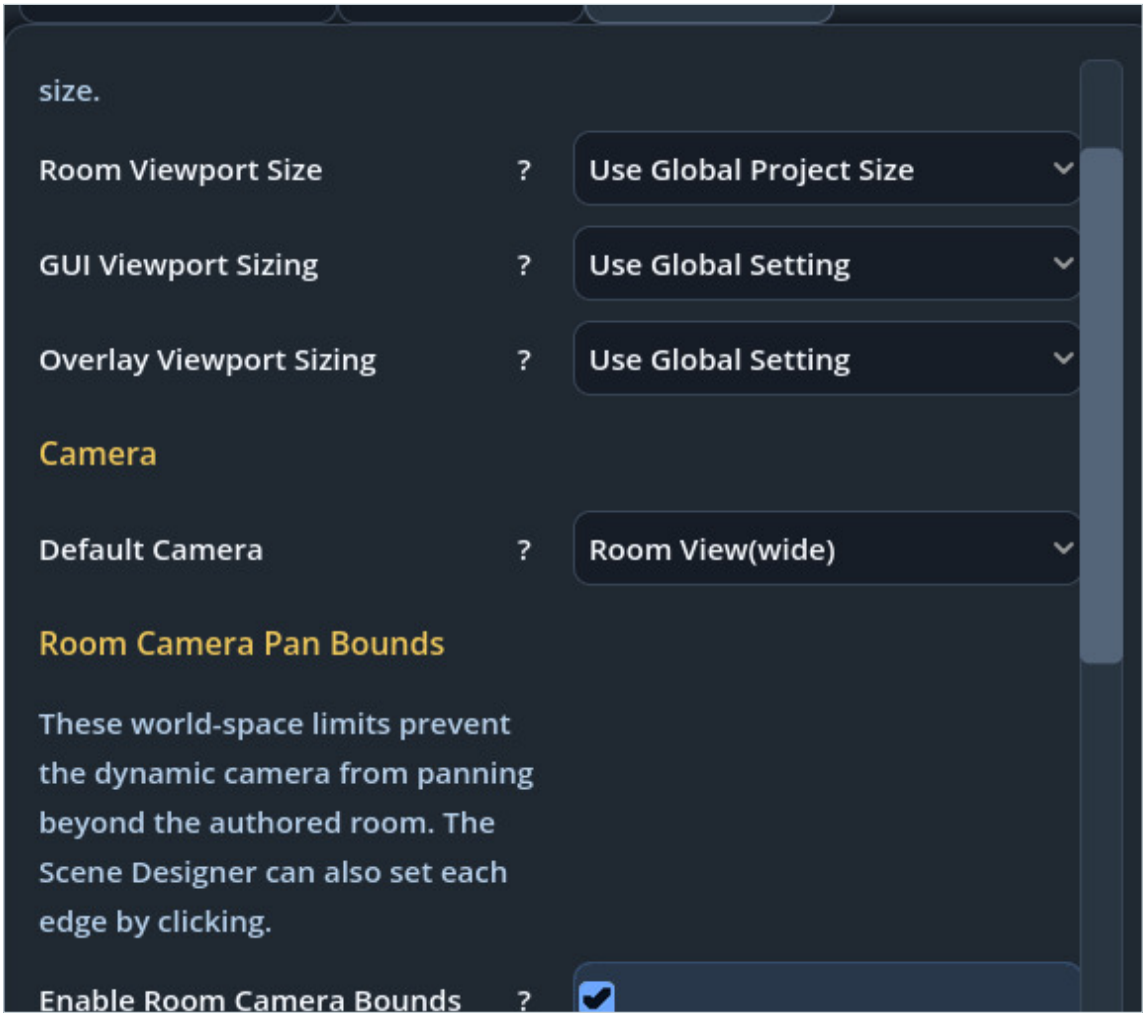


Figure 36. Room, GUI, and overlay sizing can follow different settings.

The **GUI Viewport Sizing** and **Overlay Viewport Sizing** choices decide whether those layers use inherited, independent, or room sizing. Changing the render size is not the same as enlarging the floor, background plane, camera bounds, or navigation.

After a custom size, test a wide and a narrow window. Check the GUI layout, overlay alignment, camera framing, and pointer accuracy. A view that looks right only at the editor's present panel width is not finished.

19. Connect rooms, entry events, and arrival points

A doorway needs a destination as well as an appearance. We will connect the workshop to a second room called The Courtyard. Use another room of your own if you prefer.

Add the destination before its doorway action

1. Save the workshop.
2. Add a new scene or clone a suitable scene, and give it a distinct ID, such as `courtyard`.
3. Choose its room family if it is new. Give it a floor, navigation, a usable camera, and a clear arrival area. Keep the existing player placement in the workshop; do not place the same reusable character in both rooms. The room change carries the active player to its destination.
4. Add a **Spawn** in the courtyard. Name it From the Workshop and give it ID `from_workshop`.
5. Place it on clear navigable ground, away from walls, blockers, and immediate return triggers. Set **Facing (Degrees)** to the direction the player should face on arrival.

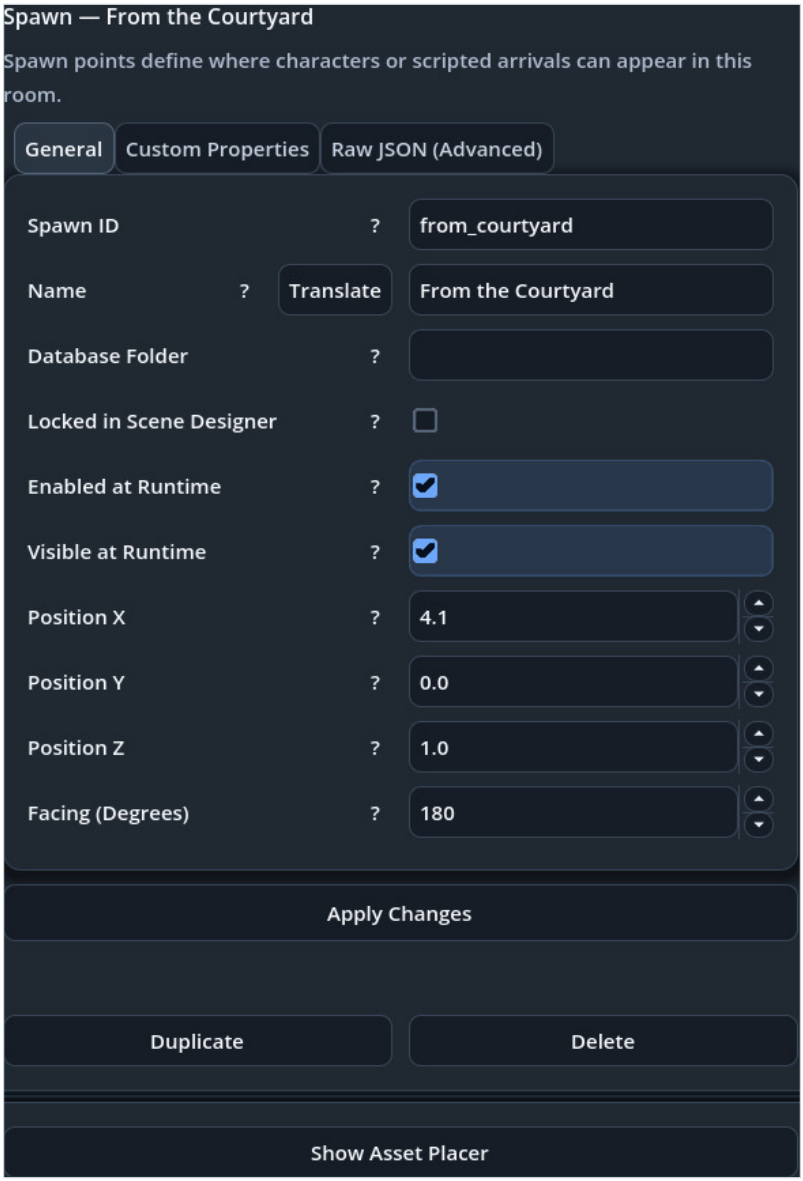


Figure 37. A spawn is an arrival position and facing. Its ID is what a room-change action references.

The pictured spawn belongs to the workshop and is named From the Courtyard. Each side needs its own clear arrival point if you want a reliable round trip.

Make the door change rooms

Select the workshop door. In **Verb Actions**, choose its Use response and set **Change Room**. Choose the destination room, its optional camera if the form offers one, and the destination spawn. Apply and test.

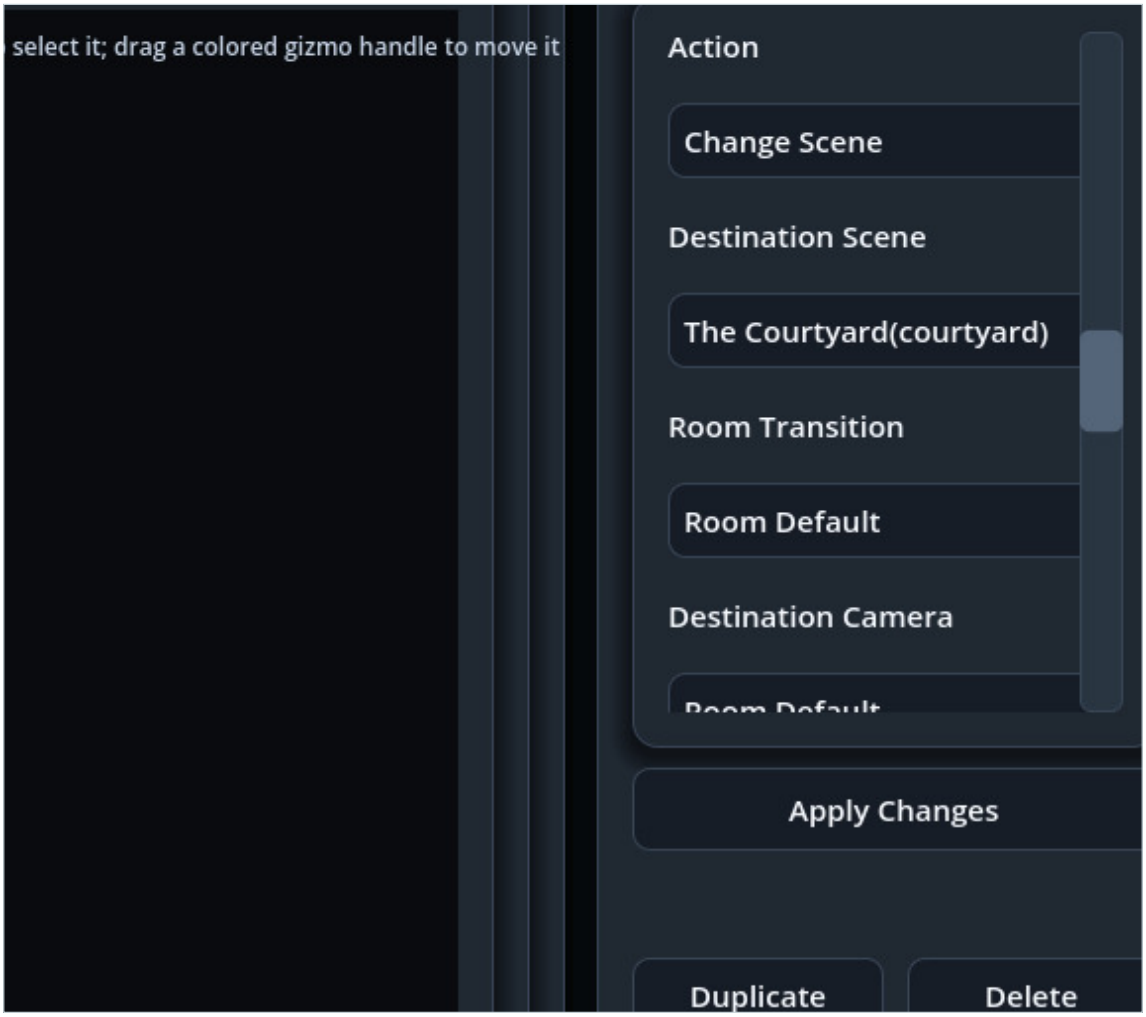


Figure 38. Change Room selects the destination and its arrival point; it can also specify the transition or destination camera.

If you keep Part 2's repair condition, put Change Room on the successful branch of that visual script. Leave the refusal on the false branch. A room-change action on the door itself bypasses that condition unless you explicitly retain the script.

Add a return door in the courtyard pointing to the workshop's `from_courtyard` spawn. Test both directions. If you arrive in the wrong place, confirm that the selected spawn belongs to the destination room, not the room you just left.

Room events

Select the room and open **Room Events**. There are four event groups:

Event	When it is useful
When Enters Room	Work that should happen on each visit.
When Enters Room First Time	An introduction that should happen once per saved game.

Event	When it is useful
When Exits Room	Cleanup or behavior whenever the player leaves.
When Exits Room First Time	A one-time departure event.

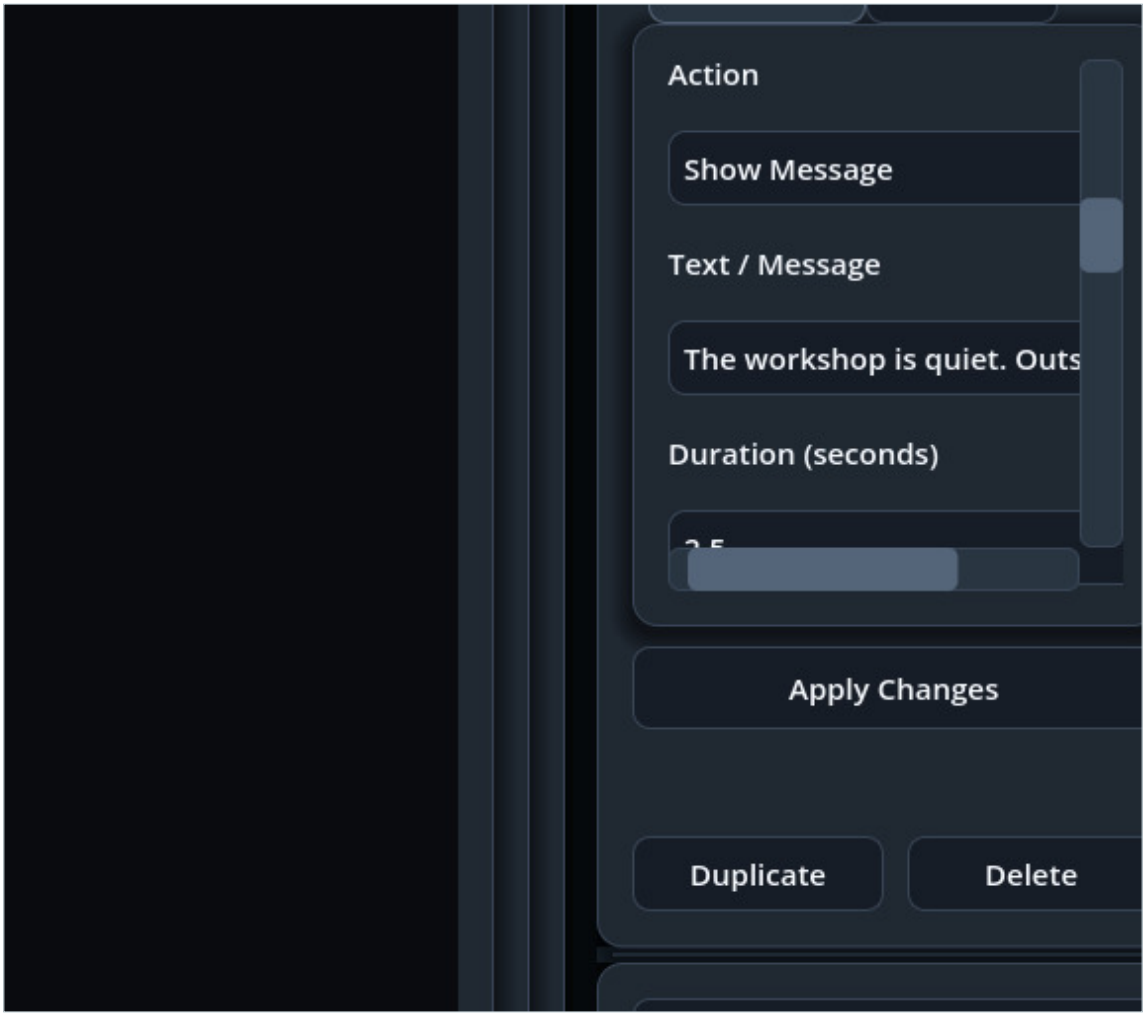


Figure 39. First-time room events are tracked per saved game, rather than per editor selection.

Add a short first-entry message as a test. Actions run in their authored order. Use **Add Next Action** for another step; choosing **No Action** and applying removes an unwanted step. Use a visual script for branching or more involved timing.

Do not place an unconditional Change Room in an entry event that returns straight to the previous room. That can create a loop before the player gets control.

Entry transitions

Open the room's **Transitions** tab. Choose the entry effect and duration. Start with a short dissolve or fade; refine color, direction, origin, and intensity only when that effect uses them.

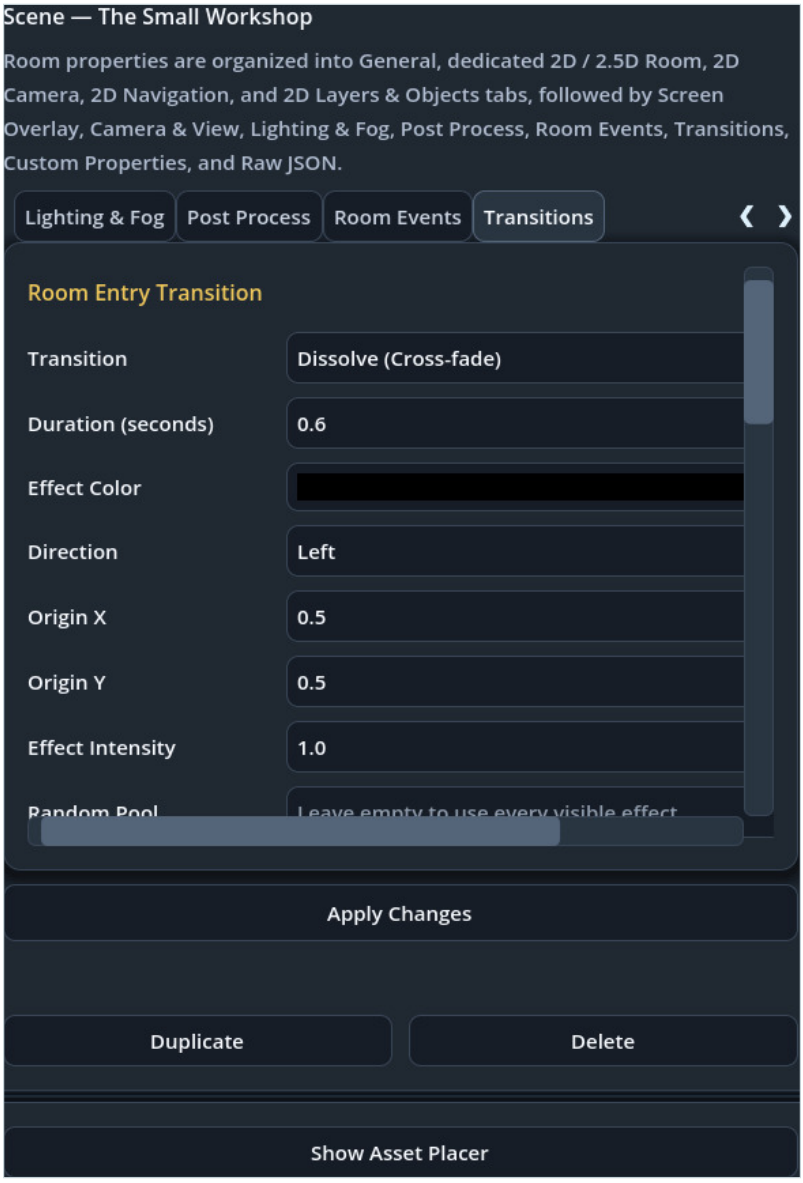


Figure 40. The room's entry transition is a presentation setting, separate from the doorway's destination.

Random Pool restricts a random transition to the IDs you supply. **Avoid Immediate Repeat** prevents the same choice immediately following itself. **Show Transition Name** is useful while testing and usually unnecessary in the finished game.

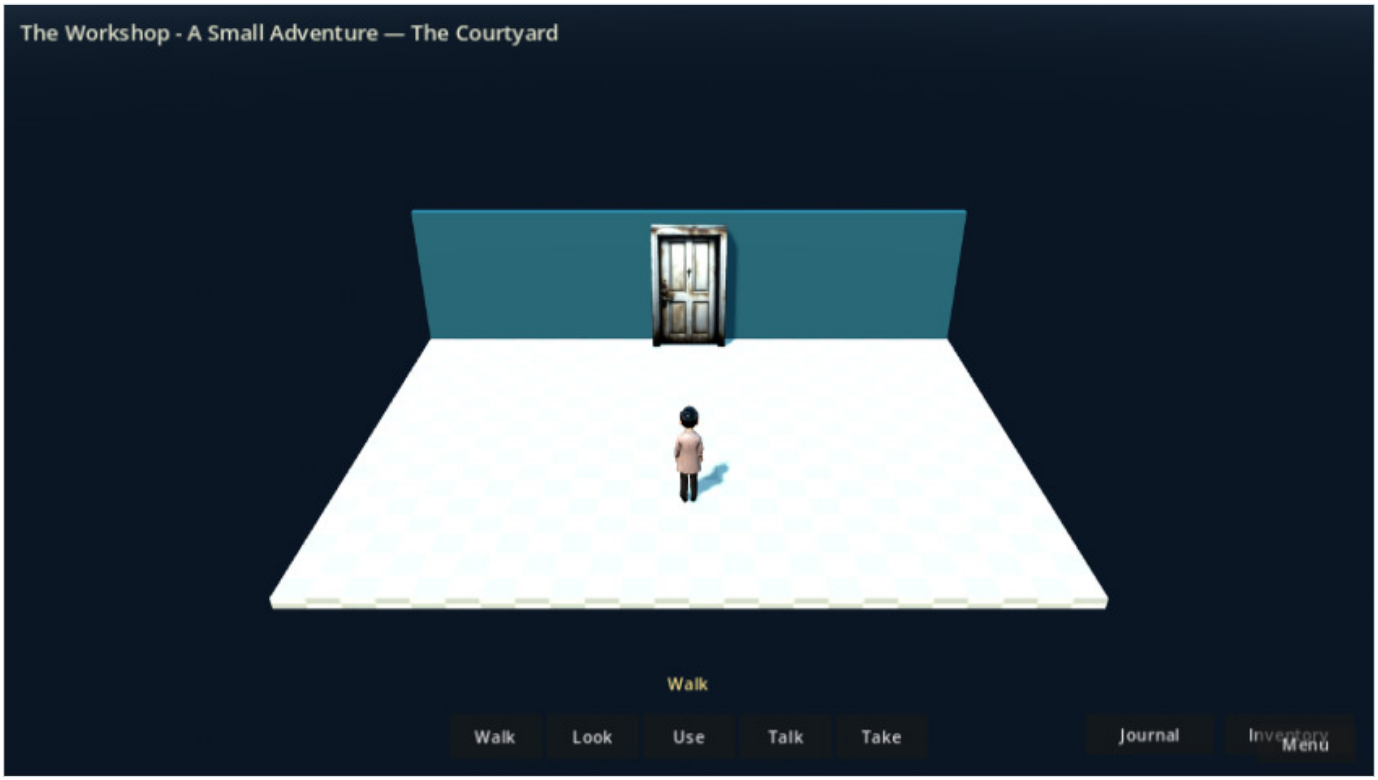


Figure 41. The doorway has loaded the separate courtyard room and placed the existing player at its destination spawn.

Run a complete round trip using normal interactions. A direct Test Scene of the courtyard proves it can start, but does not prove that the workshop door chooses the correct destination, spawn, or transition.

20. Start a 2D or 2.5D stage room

A stage room uses a flat image plane, shared world-space navigation, and one locked orthographic stage camera. The editor keeps the picture, paths, triggers, and character positions on the same stage.

The **2D Stage family** can use either the 2D or 2.5D variant. The 2.5D variant permits ordinary 3D elements as well. A room already locked to the 3D family cannot be converted into this family by changing the editor's view selector. Create a new room for the stage example.

1. Save your project and add a new scene.
2. Select the room and open **2D / 2.5D Room**.
3. Choose the required **Room Family / Stage Variant** and apply. For this example choose 2.5D.
4. Import your background as an image asset.
5. Choose it under **Background Image Asset (First Frame)**.
6. Set the plane size to match the intended world measurements and image proportions.

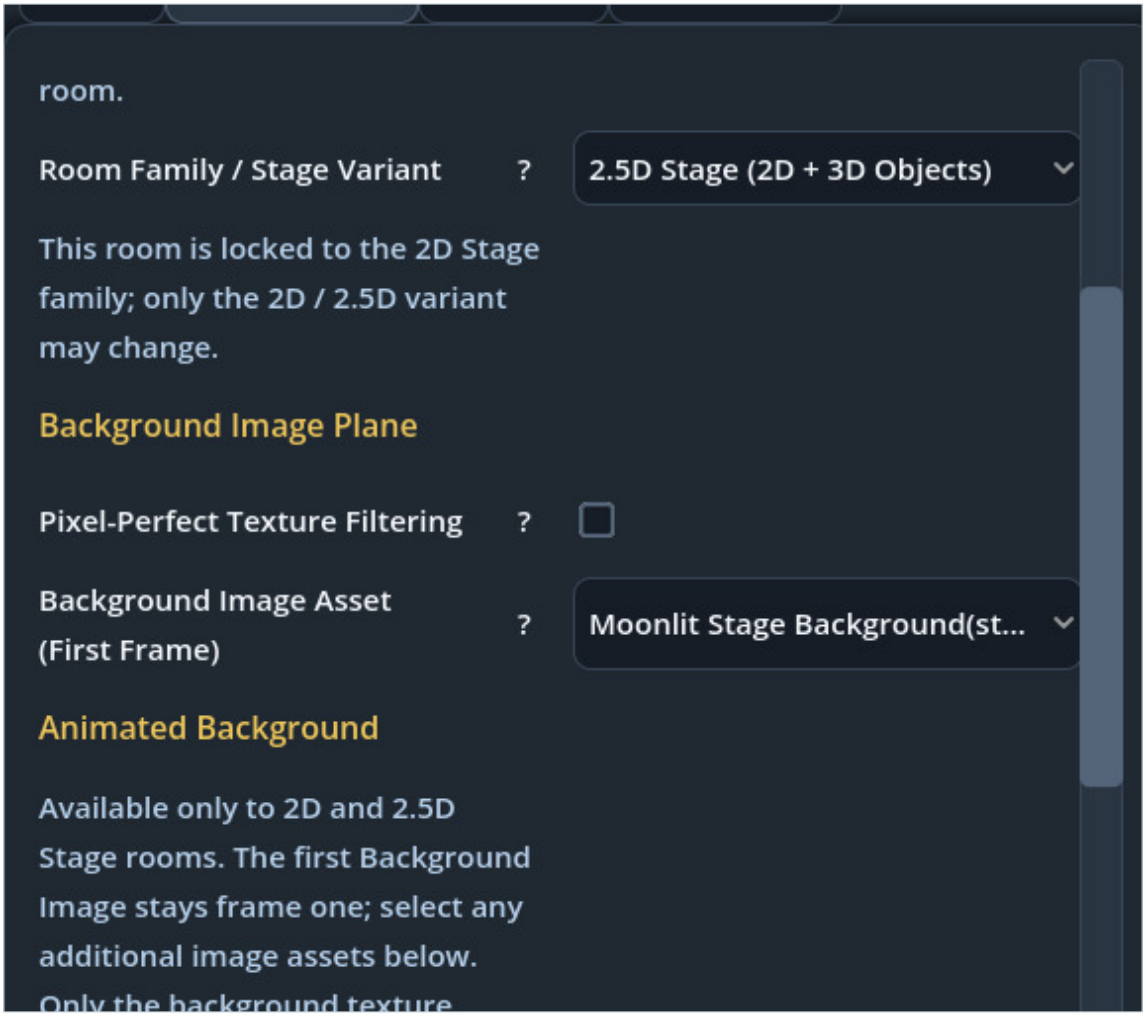


Figure 42. The stage variant belongs to the room. It is a different choice from the editing view.

The example uses a 1280 × 720 image on a 20 × 11.25 plane. That preserves the picture's 16:9 proportions. At 64 source pixels per world unit, a 640-pixel horizontal difference corresponds to 10 world units.

The stage uses X for left/right and Z for up/down across its ground plane. Y is height above that plane. The camera looks toward the plane along the stage's fixed axis. This is why a stage footprint is still described using X and Z even when the final picture looks like a conventional 2D adventure.

Change the background's size or placement

Use **Plane Width**, **Plane Height**, **Plane Position**, **Rotation Around Stage Axis**, and **Plane Scale**. Keep the image's aspect ratio unless intentional distortion is part of the design. Start with the normal plane rotation before adding a rotated background.

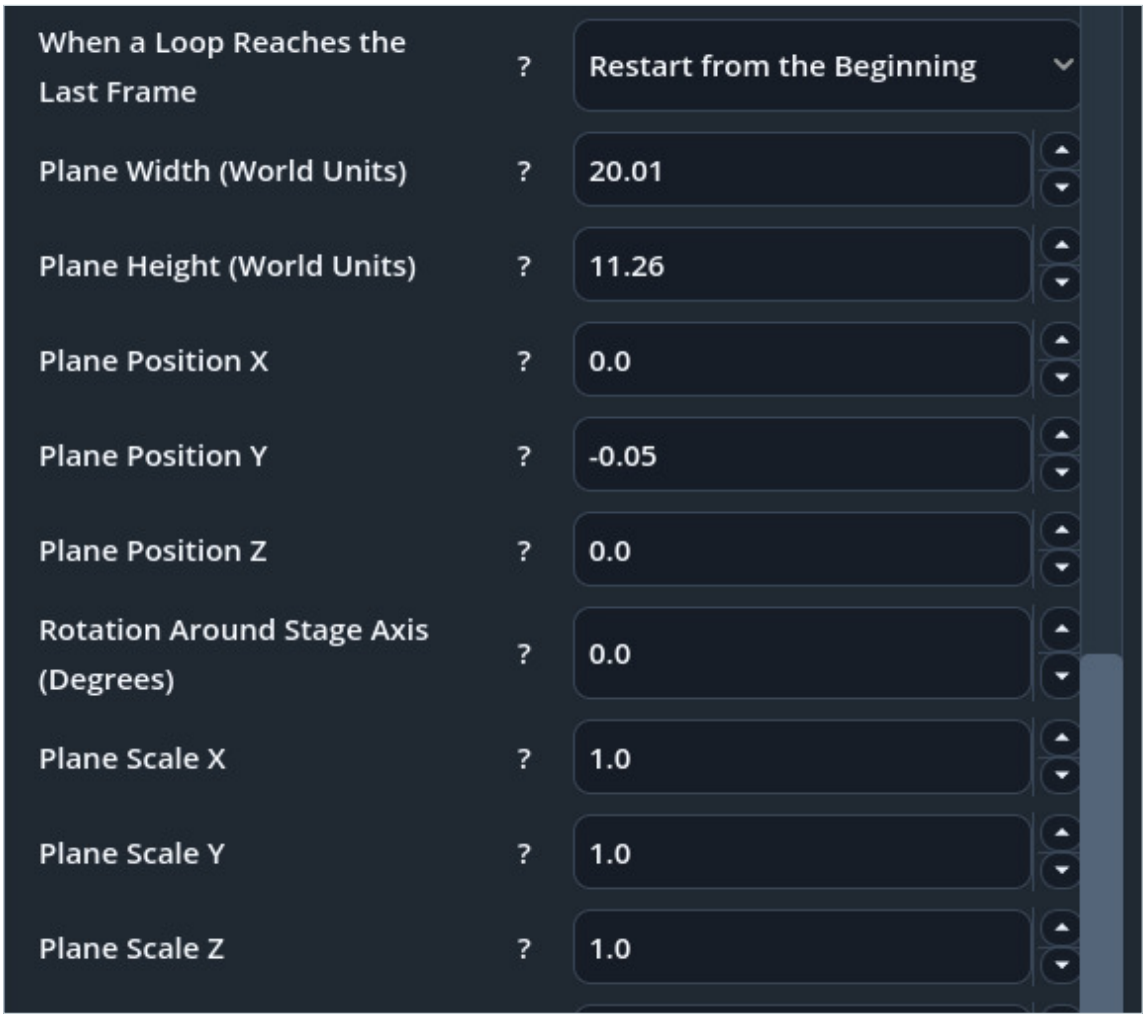


Figure 43. Plane size, source pixels per world unit, and stage ground height are related measurements, but they have separate jobs.

Pixel-Perfect Texture Filtering is useful for deliberate pixel art. Smoothly painted artwork usually benefits from ordinary filtering. This changes filtering, not the number of pixels in the original image.

You can also select **2D Stage Background** in Scene Objects to edit the background record directly. It exposes the image, size, transform, stage depth, and animation settings without requiring you to search every room setting.

Animate only the background

Enable **Animate Background** and select the additional image assets. The main background is the first frame; the additional frames follow **Asset Database Order**. Name and organize them carefully instead of assuming the order in which you clicked determines playback order.

Set **Time Between Frames**, decide whether to loop, and choose restart or backward/forward playback. All frames should have matching dimensions and alignment. Only the background texture changes: navigation, objects, triggers, characters, and scripts continue running.

This is useful for a flickering sign, a painted water cycle, or a subtle animated room background. It is not a replacement for a character's own animation states.

21. Make a stage larger and let its camera scroll

When a background extends past the useful view or the player stops before reaching its edge, do not begin by enlarging the visible grid. There are several separate sizes to check.

Setting	What it controls
Background plane width and height	How large the image is in the world.
Navigation Area outline	Where the player may walk.
Blockers and object collision	Places the player must avoid within that area.
Stage camera view height	How much of the stage appears at once.
Stage camera bounds	How far the view can reveal as it follows.
Grid spacing	The editing guide and snap increments.

Fit a static stage view first

1. Select the room's **2D Camera** tab, or select **2D Stage Camera** in Scene Objects.
2. Set **Room Scrolling** to **Fixed**.
3. Choose the view centre X and Z.

4. Set **Orthographic View Height** to show the intended portion of the picture. Smaller shows less and feels closer; larger shows more.
5. Keep camera roll at zero until the room works normally.
6. Test the actual game view and its aspect ratio.



Figure 44. Orthographic View Height changes the visible stage area. Camera Height / Distance does not work like perspective zoom.

The stage owns one main camera. Its identity, projection, viewing axis, and lock state are managed by the stage. You may change its framing, view height, roll, and following behavior. It cannot orbit away from the background as a free 3D room camera can.

Enlarge the playable area

Suppose your picture needs a 40 × 22.5 world-unit stage instead of 20 × 11.25.

1. Set the background plane to 40 × 22.5, preserving its proportions.
2. Select the relevant Navigation Area and enlarge its bounds or edit its polygon to cover the actual floor in the larger picture.

3. Move any hotspots, blockers, regions, spawns, and characters so they line up with the resized picture. Resizing the background is not a promise to scale every other record automatically.
4. Expand the stage camera bounds to the new useful extent.
5. Choose a view height smaller than the whole image if you want scrolling.
6. Walk to all four intended edges in Test Scene.

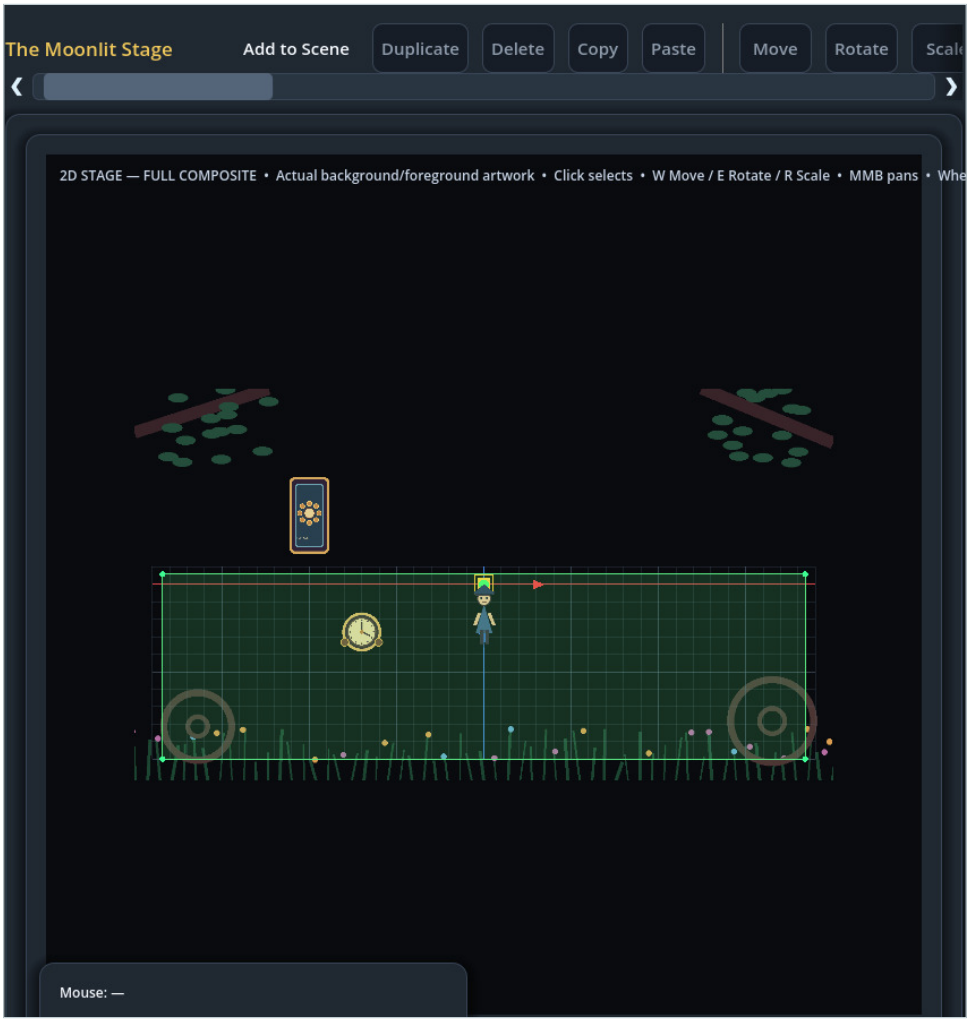


Figure 45. The navigation polygon is edited over the background. Its vertices determine where the player may walk.

For a drawn street scene, the entire picture is rarely walkable. Draw around the ground and exclude buildings, scenery, water, and other unreachable areas. Keep the character's collision radius in mind near narrow paths.

Follow the player

Choose **Follow Player** for continuous scrolling, or **Follow Near Screen Edge** to wait until the player reaches the dead zone. Choose horizontal, vertical, or both axes. Start with a modest dead zone, smoothing enabled, and follow speed around 5, then test.

Look-Ahead X/Z biases the framing. Bounds prevent the view revealing blank space beyond the artwork. A view as large as the whole background has little useful space left to scroll, even when Follow is enabled.

The stage camera's Advanced Camera settings provide the additional follow tools where applicable, while preserving its stage viewing axis. Start with ordinary stage follow before combining systems. If movement stops early, check navigation and collision. If only the picture stops scrolling, check the camera's size and bounds.

22. Add depth with parallax and walk-behind scenery

Parallax changes how image layers move with the camera. A walk-behind changes which parts of the character are covered by scenery. They solve different problems.

Add a parallax plane

1. Select the room and open **2D Layers & Objects**.
2. Click **Add Parallax Plane**, or use the appropriate 2D background-layer option in Add to Scene.
3. Select the new record under **Parallax Planes** in Scene Objects.
4. Choose the image and set its world size, position, and layer order.
5. Adjust **Horizontal Factor** and **Vertical Factor**. Start around 0.5 for a distant layer and compare it with 1 in a running scrolling view.
6. Walk far enough for the camera to move. Refine the factors by watching the layers relative to each other.



Figure 46. Each parallax plane has its own image, transform, order, and movement factors.

A parallax plane behind a completely opaque background may be working perfectly and still be invisible. Give the foreground/background artwork transparent openings or arrange the layers so the rear image can be seen. In the layered example, the distant garden supplies the rear picture and the transparent leaves supply the foreground artwork.

Make the rear image large enough to cover the view throughout the camera movement. A perfect initial composition can still reveal an empty strip after scrolling.

Make an arch cover part of a character

1. In **2D Layers & Objects**, click **Add Walk-Behind Layer**.
2. Select the walk-behind in Scene Objects.
3. Assign a transparent cutout image, such as the garden arch, and place/size it over the scene.
4. Set its footprint polygon where the occlusion should matter. Use the stage Designer's shape controls for visual editing.
5. Set **Baseline (World Z)** to the line that separates the front and back of the scenery.
6. Walk the character across that line and back while overlapping the artwork.

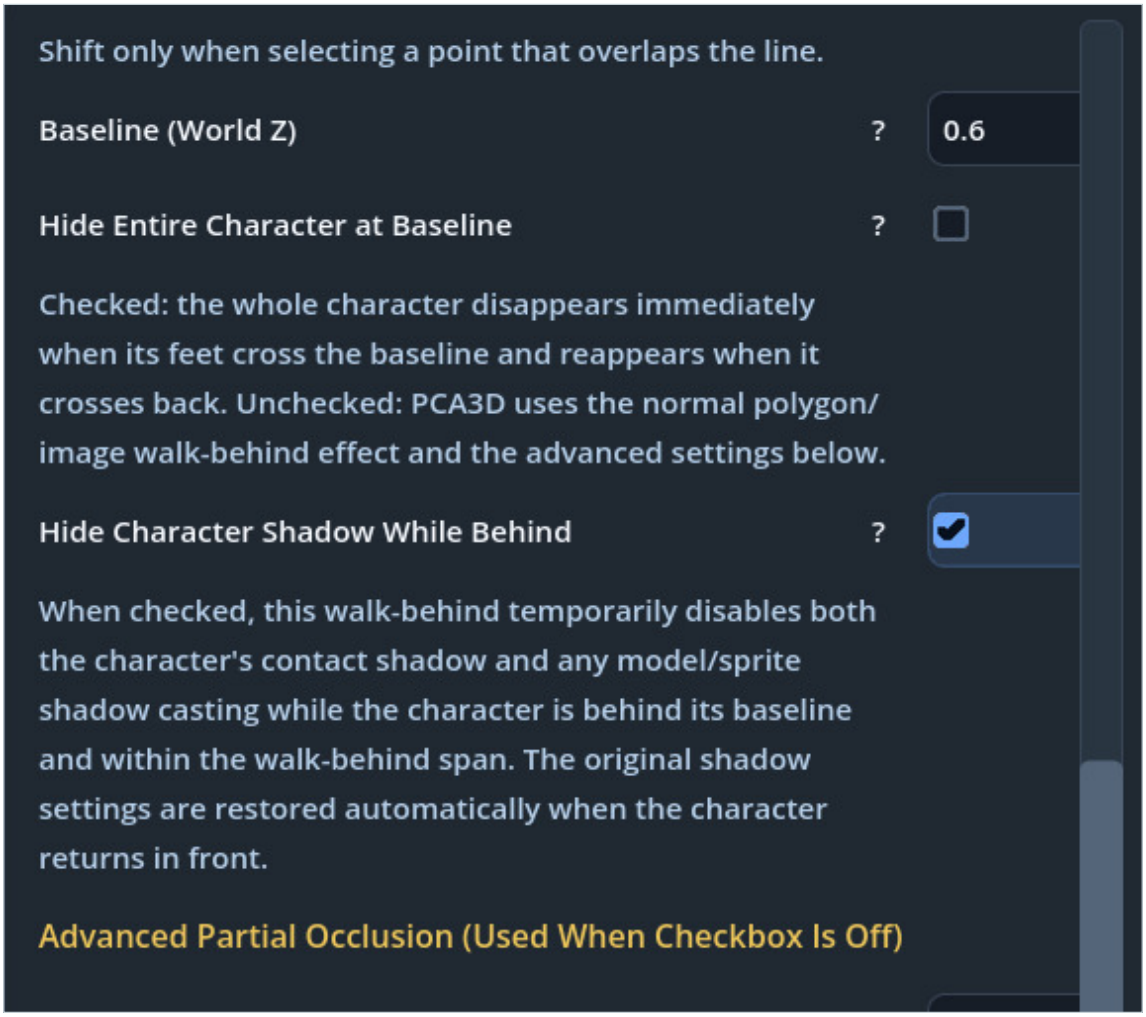


Figure 47. The polygon describes the relevant footprint; the pink baseline controls which side counts as behind.

Drag the pink line or its centre handle to move the baseline. Hold Shift when selecting a vertex that overlaps the line. The numerical polygon field is available when you need exact `[[x,z], ...]` points.

With **Hide Entire Character at Baseline** off, the normal image/polygon effect can cover only the appropriate parts of the character. With it on, the whole character disappears as its feet cross the baseline and reappears on the return. Use the full-hide option for an actual offscreen or hidden space, not merely because an arch has two posts.

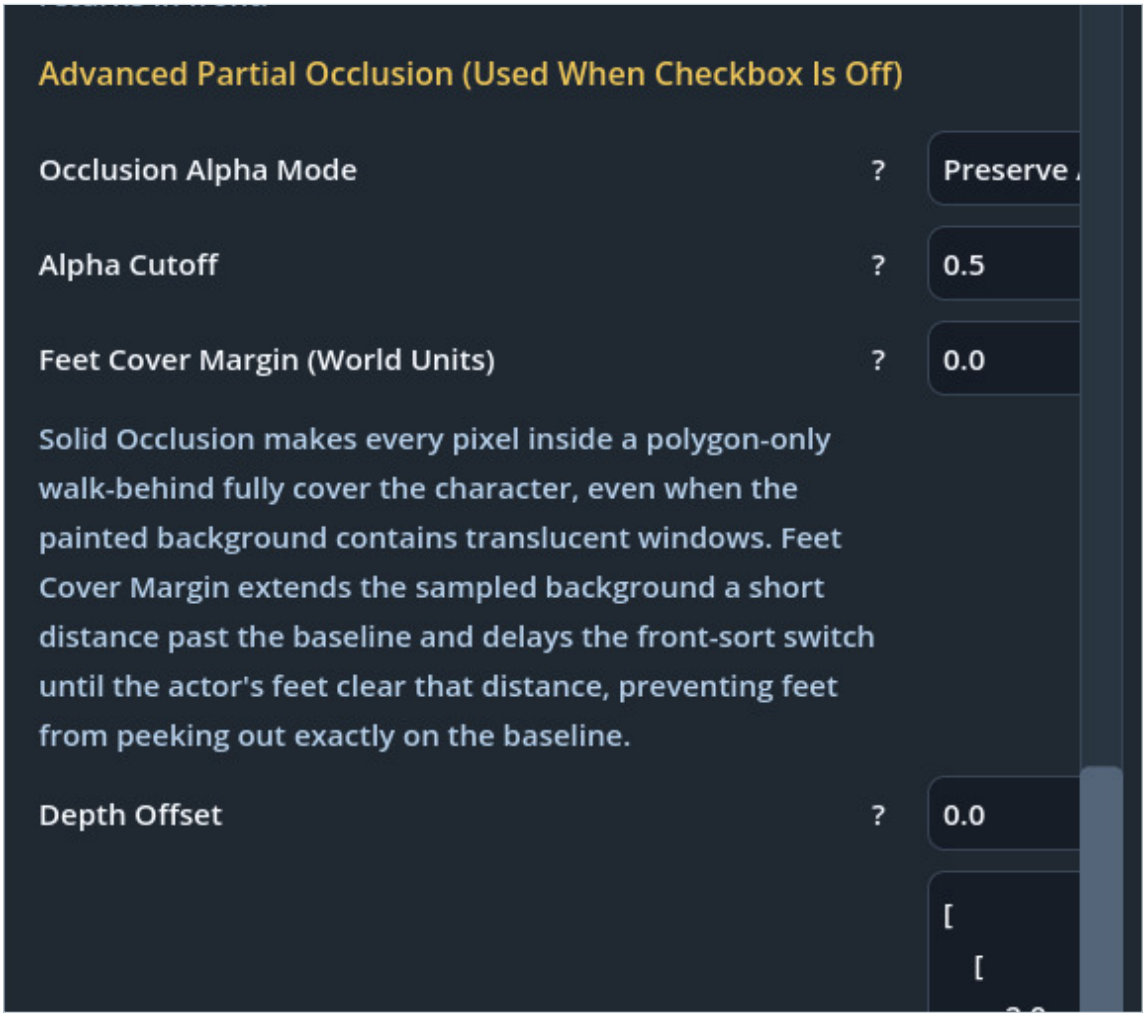


Figure 48. The advanced partial-occlusion controls are used when full-character hiding is off.

Preserve Alpha respects transparent artwork. **Alpha Cutoff** treats pixels above the threshold as solid. **Solid** gives an opaque mask. **Feet Cover Margin** helps avoid a visible sliver at the feet; **Depth Offset** adjusts the layer relationship. Use small values and test both sides of the baseline.

You can leave the walk-behind image empty and use the polygon to clip the matching portion of the stage background. This lets painted foreground scenery cover the character without importing a separate cutout. A separate transparent image is easier to reason about when the art is already available.

Hide Character Shadow While Behind prevents a shadow from giving away a character meant to be hidden. Check it with the actual character and lighting rather than assuming the image alone hides the shadow.

23. Place sprites, billboards, and stage characters

The stage uses ordinary world objects for placement and interaction. You can move, scale, and rotate them using their normal **Visual Adjustments** controls. Their presentation settings are collected in **2D / 2.5D**.

Add a picture as an object

1. Import the picture as an image asset. Use transparency around the visible object where appropriate.
2. Choose a sprite/billboard entry in **Add to Scene > 2D / 2.5D**, or select an ordinary scene object that should show the image.
3. In **2D / 2.5D**, choose the image asset.
4. Set **Stage Rendering** to the camera-facing world billboard treatment when that is the intended result.
5. Choose the plane orientation and billboard rotation behavior.
6. Apply and inspect its position from the stage camera.

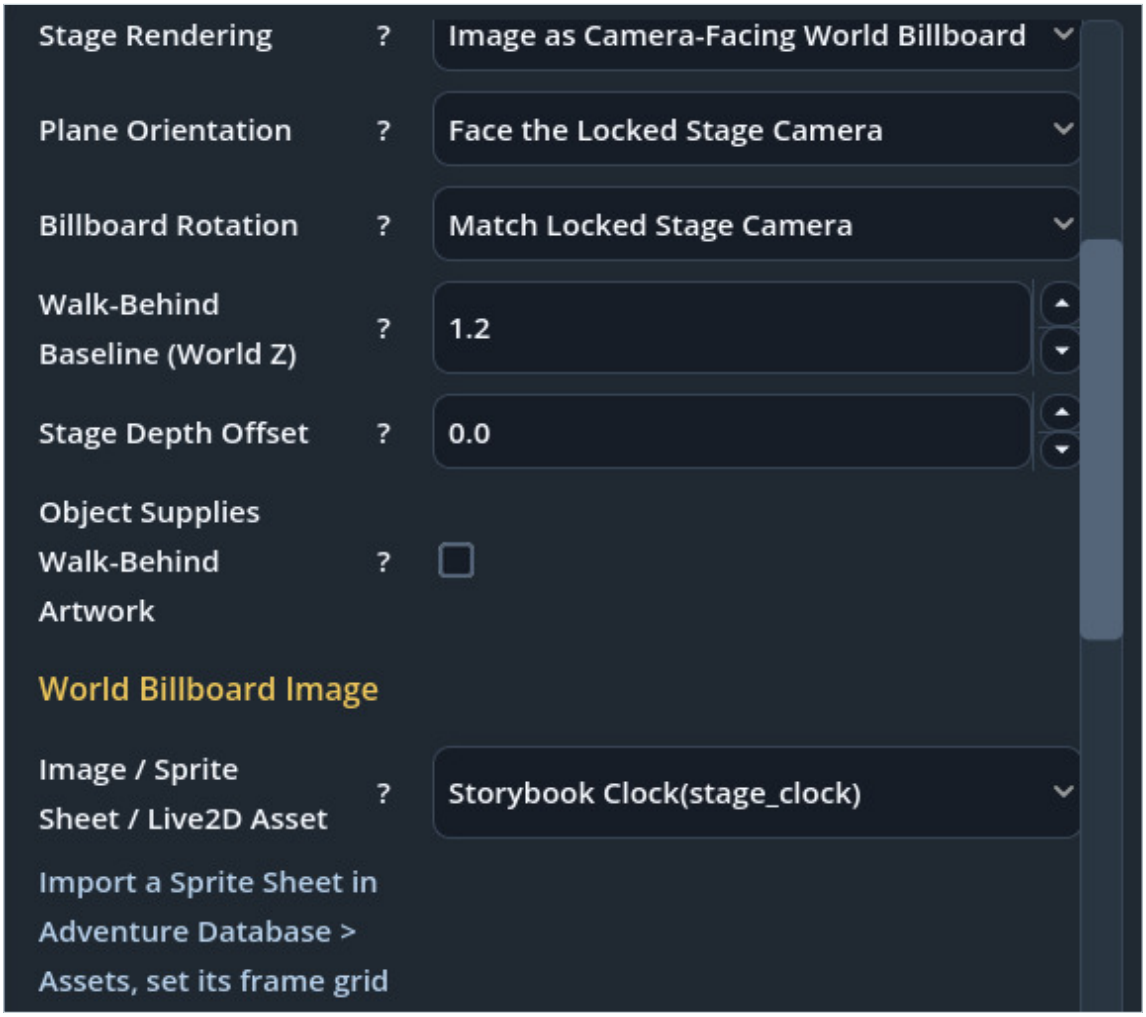


Figure 49. Stage presentation controls how an image plane faces the camera and participates in stage depth.

Plane choice	Use it when
Face the Locked Stage Camera	The image should read correctly in the stage's fixed view.
Lie Flat on the Background Plane	It is a floor marking or artwork meant to lie on the stage.
Stand Perpendicular to the Stage	It should stand upright relative to the stage.
Use Authored 3D Rotation	You intend to control the plane's orientation yourself.

Billboard rotation can match the locked stage camera, always face the active camera, or use no automatic rotation. Do not fight an automatic facing mode by repeatedly changing the object's rotation. Choose the correct mode first.

Walk-Behind Baseline, **Stage Depth Offset**, and **Object Supplies Walk-Behind Artwork** control the object's relationship to other stage content. They are not replacements for a physical collision box or a hotspot's event volume.

Give a sprite character the right size and feet

Create the reusable character through Character Designer, then place that character in the room. For a sprite-sheet character, use **Animation States** with source **Sprite Sheet Asset**. The sheet's frame grid and FPS belong to the imported asset/state configuration. Do not use the whole sheet as a single portrait of sixteen tiny characters.

The placed character's stage settings include **Fit Character to Visible Pixels**, **Visible Character Height**, and **Foot Anchor Offset**.

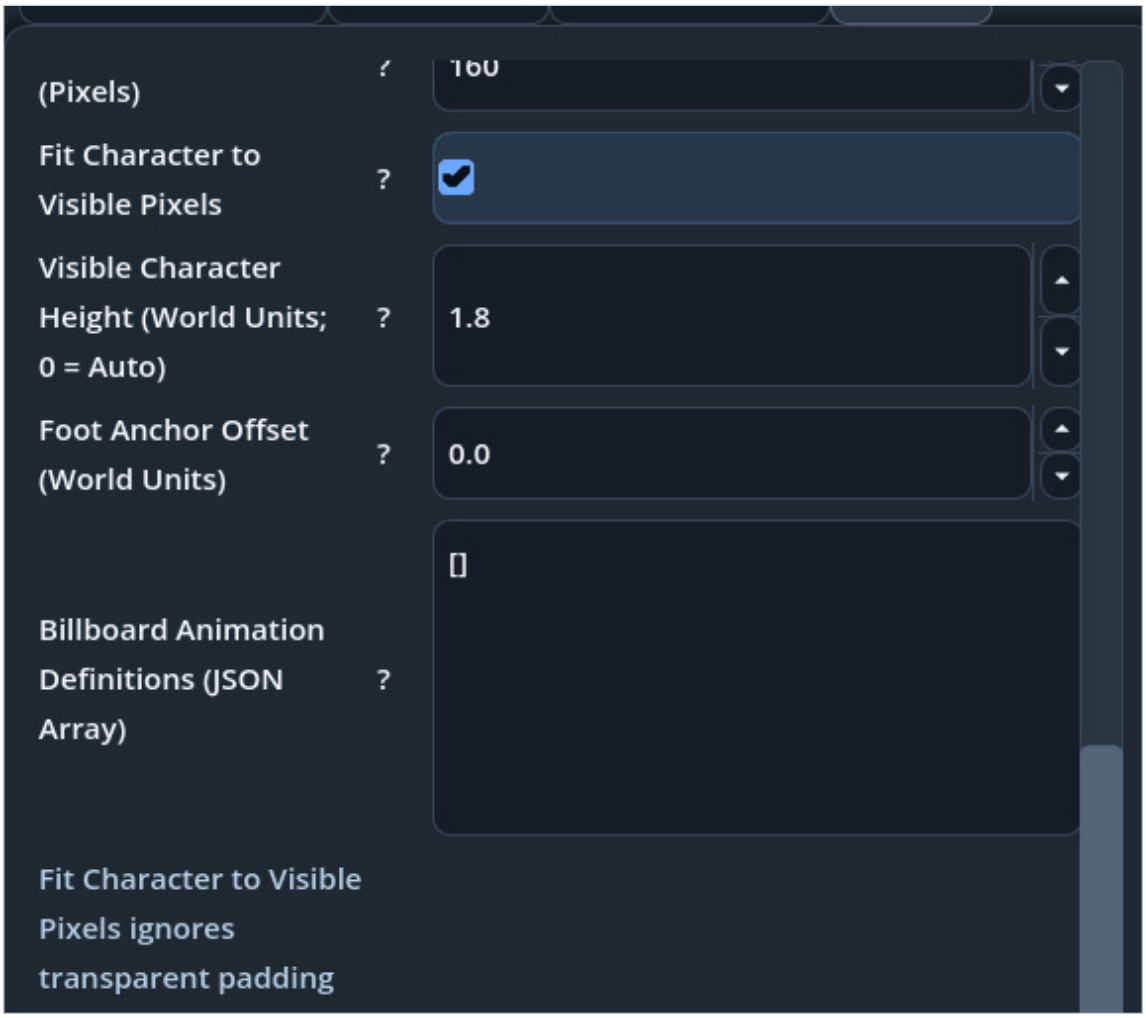


Figure 50. These controls help a padded sprite frame stand at the intended height with its feet on the stage.

Leave automatic fitting on to ignore transparent padding. Set a visible height when the automatic result needs correction. Adjust foot offset only after the height is right. A large transparent border can otherwise make a character look too small or float above the floor.

For an object billboard, the same page supplies a default FPS, looping choice, default animation state, and animation-definition data. For a character, keep the authored idle/walk/talk states in the character animation tools so normal gameplay can choose them.



Figure 51. A running stage view shows the character, background, and placed artwork together.

Ordinary 3D assets in 2.5D

A 2.5D stage allows ordinary 3D models, lights, and effects while keeping the stage camera. Add the model using the asset workflow from Part 2. Use normal XYZ transforms and a fitted collider.

The fixed camera may look down on a model from an angle that does not show its interesting face. Rotate or position the asset appropriately, or use a billboard image when the art is meant to read as a flat drawing. Switching to a free editing view does not change the stage camera's game view.

24. Finish the stage's interactive parts

The stage shares the same navigation areas, spawns, triggers, regions, lights, particles, and action system as the 3D workflow. You do not need to invent a second set of puzzle rules for a painted background.

Add to Scene choice	What to configure next
2D Hotspot	The interactive object's name, event volume, and verb responses. Use an empty interactive object when no separate visible art is needed.
2D Trigger	The event volume, entry/exit or pointer events, and optional verb actions.
2D Region	The footprint, event height, actions, and optional effects.
2D Navigation Blocker	The part of the stage that must remain out of the walking route.
2D Point Light	The light's world position, color, intensity, and range.
2D Particles	The emitter's preset, placement, motion, and scale relative to the stage.
Foreground / Walk-Behind Layer	The artwork, depth, baseline, and occlusion settings.
2D Character	The reusable character and its sprite animation setup.

Do not assume that the 2D menu label changes every measurement into source-image pixels. The current stage stores shared world-space placement and navigation. Read field labels, use the world-size relationship from Chapter 20, and test the result.

Clean 2D Room

Enable **Clean 2D Room** to hide editing helpers and categories that get in the way of judging the artwork. It is available for stage rooms, not ordinary 3D rooms. The editor can reveal a selected or newly created record even while this display mode is active.

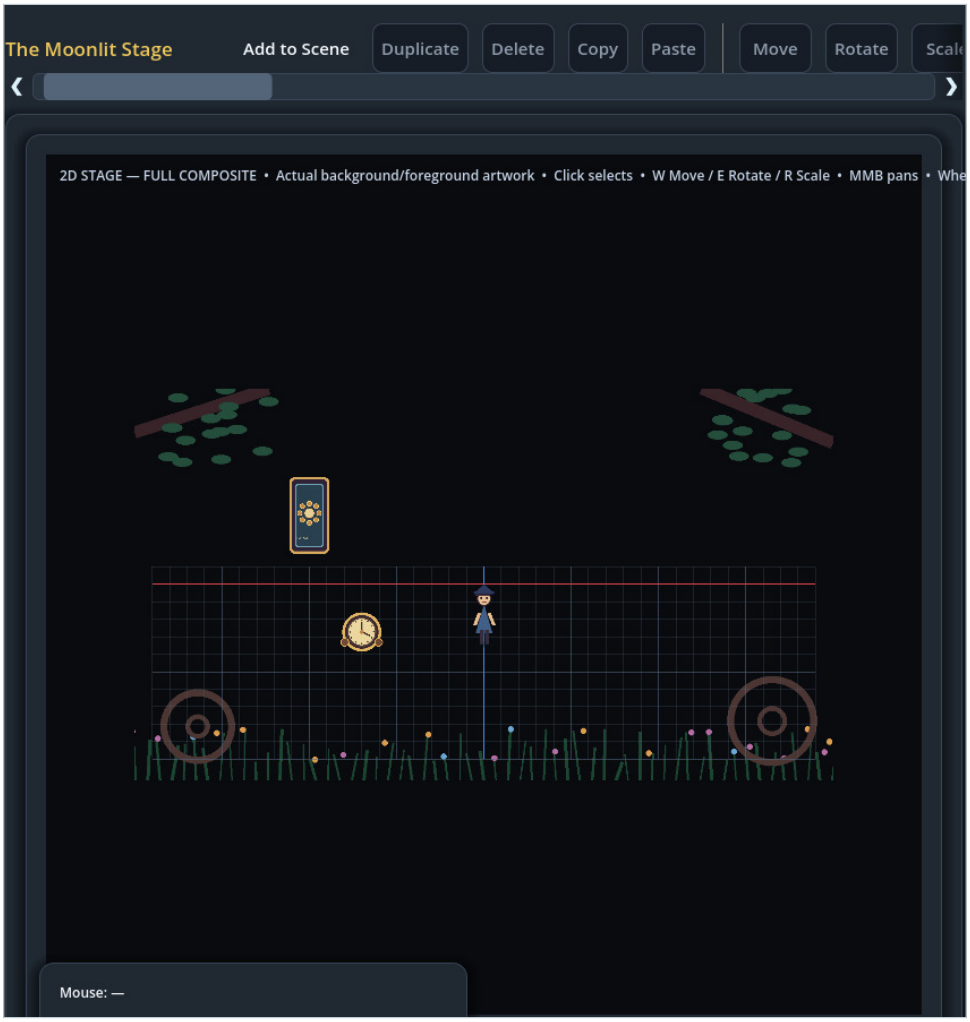


Figure 52. Clean 2D Room reduces editing clutter; it does not disable gameplay records.

This is an editor display choice. It does not disable hidden helpers at runtime. Turn it off again when you need to inspect paths, triggers, regions, cameras, or other supporting records.

Test the whole picture

Walk across the stage, behind the foreground art, and to the edges of the allowed ground. Test hotspots both before and after the camera scrolls. Check sprite size, baseline crossings, shadow behavior, and layer seams. Stop, correct one issue, apply, and repeat.

A flat scene can have sophisticated behavior, but its success still comes down to a simple test: does the character stand, walk, and interact where the picture tells the player it should?

25. Use the remaining procedural scenery

Procedural objects give you recognizable scenery without importing a model for every chair, fence, or rock. They are particularly useful for blocking out a scene and checking its proportions before committing to final artwork.

1. Choose an item from **Add to Scene > Procedural**.
2. Select the new object and give it a meaningful name.
3. In **General**, check **Type**, **Generator**, and **Size**.
4. Use **Visual Adjustments** for position, rotation, and transform scale.
5. Use **Visual Effects** for color, texture, roughness, and other material treatment.
6. Check its collider and blocking before testing the character's route.

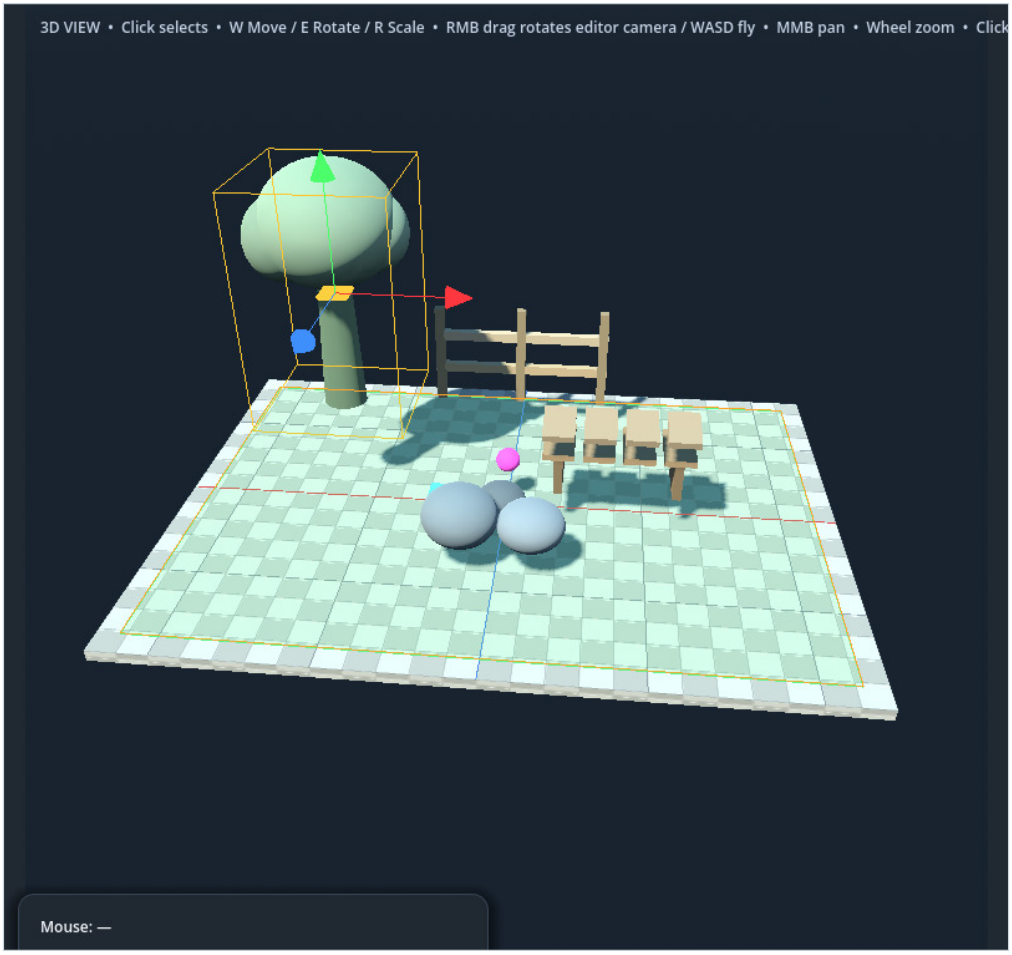


Figure 53. A tree, fence, bench, and rocks made with the built-in generators.

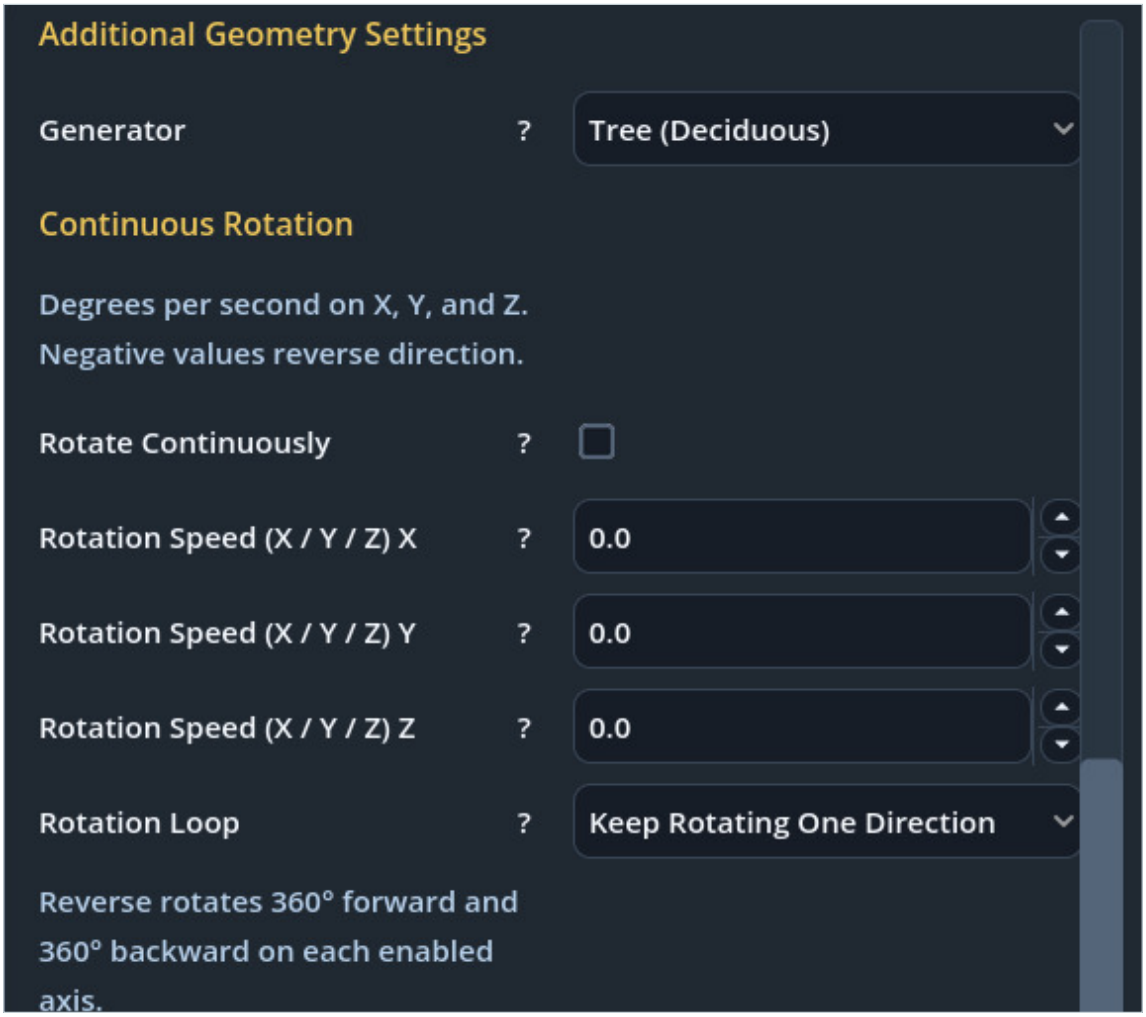


Figure 54. Procedural scenery uses the same placement and material workflow as the workshop's first generated table.

Group	Available generator choices
Furniture and room structure	Table, square table, rectangle table, bookcase, bookcase with books, floor, wall, chair, lamp.
Doors	Door, panel door, glass door, double door.
Windows and rugs	Window, arched window, sliding window, bay window, round rug, square rug.
Trees and ground detail	Tree, pine tree, dead tree, log, tree stump, rock, boulder cluster, bush, hedge, flower patch.
Outdoor furniture	Fence, fence gate, park bench, picnic table, signpost, lamp post, campfire, tent.
Street pieces in the Generator selector	Road section, sidewalk, curb, street lamp, traffic cone, fire hydrant, mailbox, dumpster, traffic light, bus stop.

The direct Add menu contains the main presets; the General page's Generator selector includes additional choices such as the street pieces. If you cannot find a direct menu entry, add a procedural object and choose the generator there.

Size changes the generated geometry. Transform scale changes the placed object's scale. A custom collider has its own local dimensions and follows the transform. These measurements may be related, but they are not interchangeable.

Simplify Generated Geometry is available for generated primitives/procedural content where supported. It is an optional detail reduction, not an instruction to rewrite imported animated models. Test the silhouette and shadows after using it.

A generated door is scenery until you give it an interaction. A generated campfire can provide a useful shape, but check whether you also need a light, a particle emitter, ambient audio, and actions. The name of a generator does not guarantee a complete gameplay system.

26. Custom properties, Raw JSON, and movie scenes

These are the remaining advanced data tools exposed through the Scene Designer's Inspector. They are useful, but none is required to place a normal prop or make a simple conversation.

Store a value on one record

Select an object and open **Custom Properties**. Click **Add Property**, give it a stable ID and a readable display name, choose its type, and enter a value.

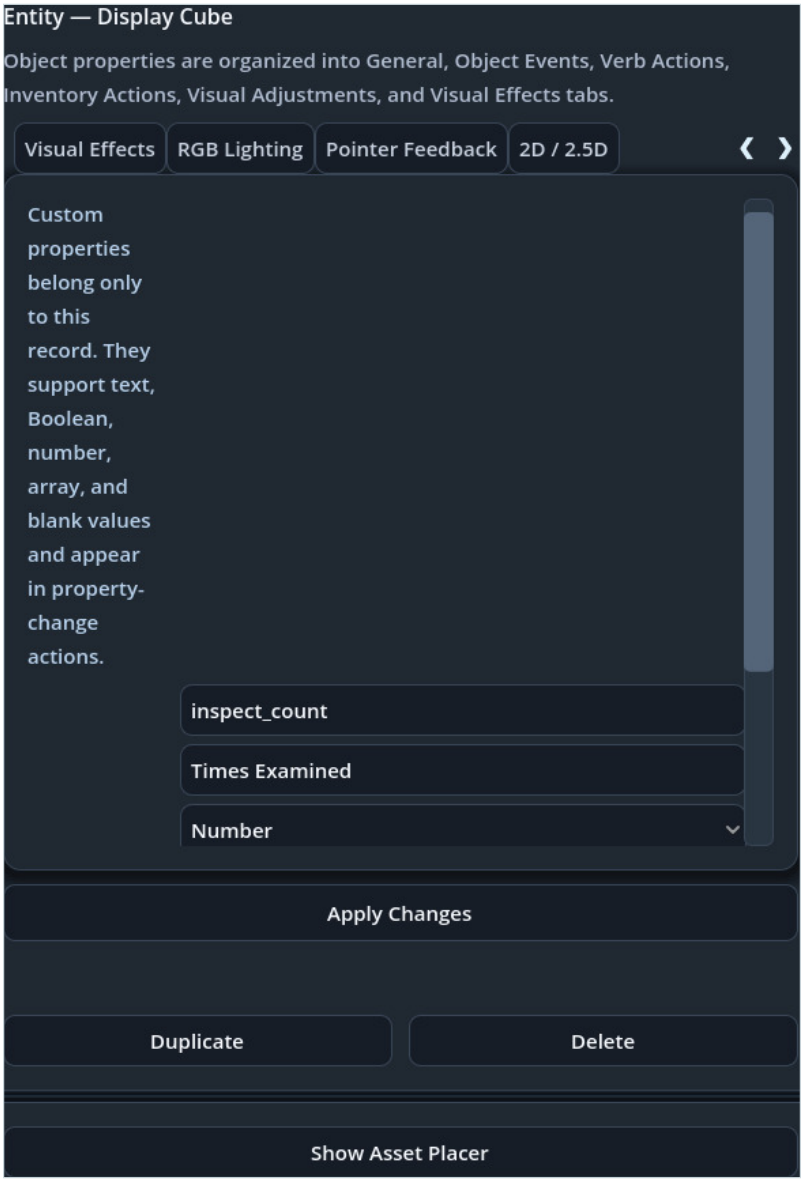


Figure 55. A custom property belongs to the selected record. The example stores a number called Times Examined.

For example, `inspect_count` can be a number starting at zero. A `powered` property could be Boolean. A text property could hold an object's inscription. An array holds a list, entered as valid JSON; **Blank / null value** represents an intentionally empty value.

A custom property does not do anything until an action or script reads or changes it. Use the appropriate property-change action and choose the property on the correct owner. Do not confuse a record-local property with the global repair variable used in Part 2.

Raw JSON is the saved structure

The **Raw JSON (Advanced)** tab exposes the selected record's nested data. It is useful for checking an exact navigation value, a point array, or a field not exposed by the current form.

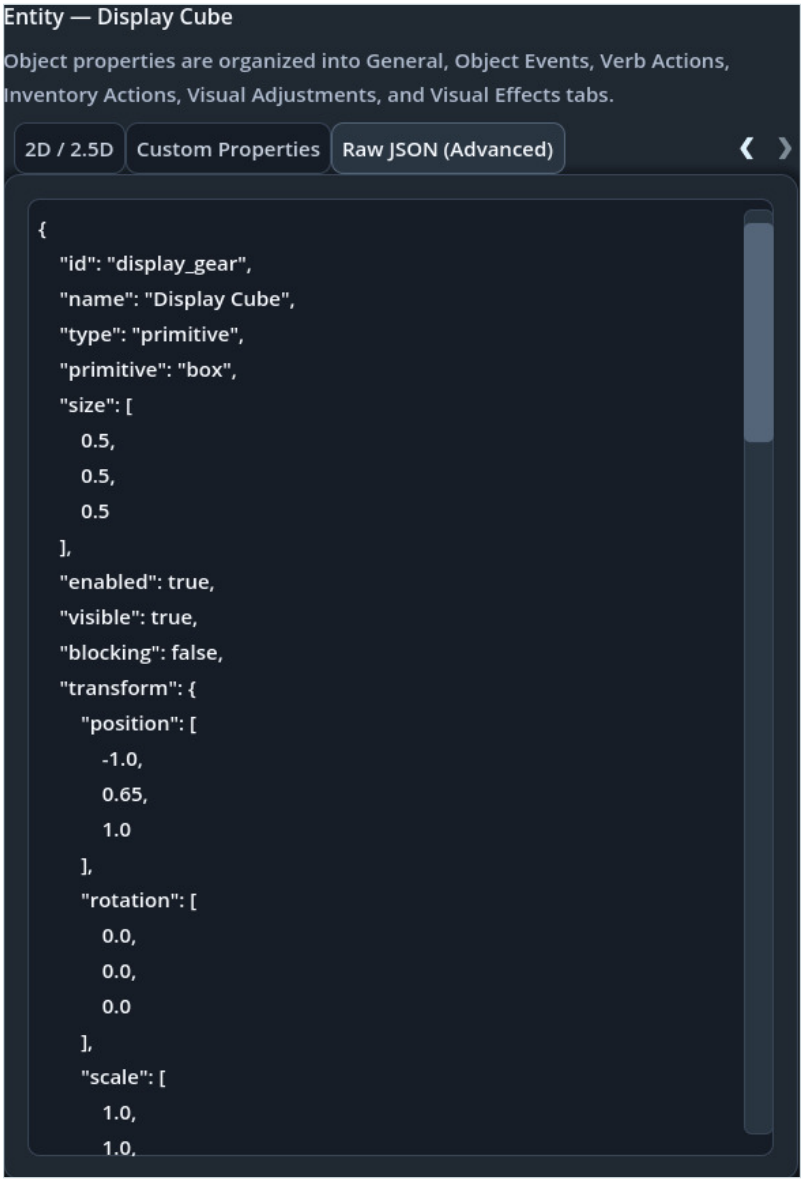


Figure 56. Raw JSON shows the selected record's structure. Use its own validation-and-apply button after an intentional edit.

Save first. Keep quotation marks, commas, brackets, and value types intact. `true` is a Boolean; `"true"` is text. Editing an ID can break references elsewhere. Prefer the normal form when it offers the field you need.

The button is **Validate and Apply JSON (Advanced)**. JSON validation checks the structure; the project's **Validate** command checks broader references and content. Neither command can decide whether your camera looks good or whether a puzzle is enjoyable.

Movie / FMV scene controls

The room Inspector also includes **FMV Scene**, **Movie Asset ID**, autoplay, mute, start time, and **Scene After Movie**. In this version these additional fields sit below the transition and gravity controls in **Transitions**.

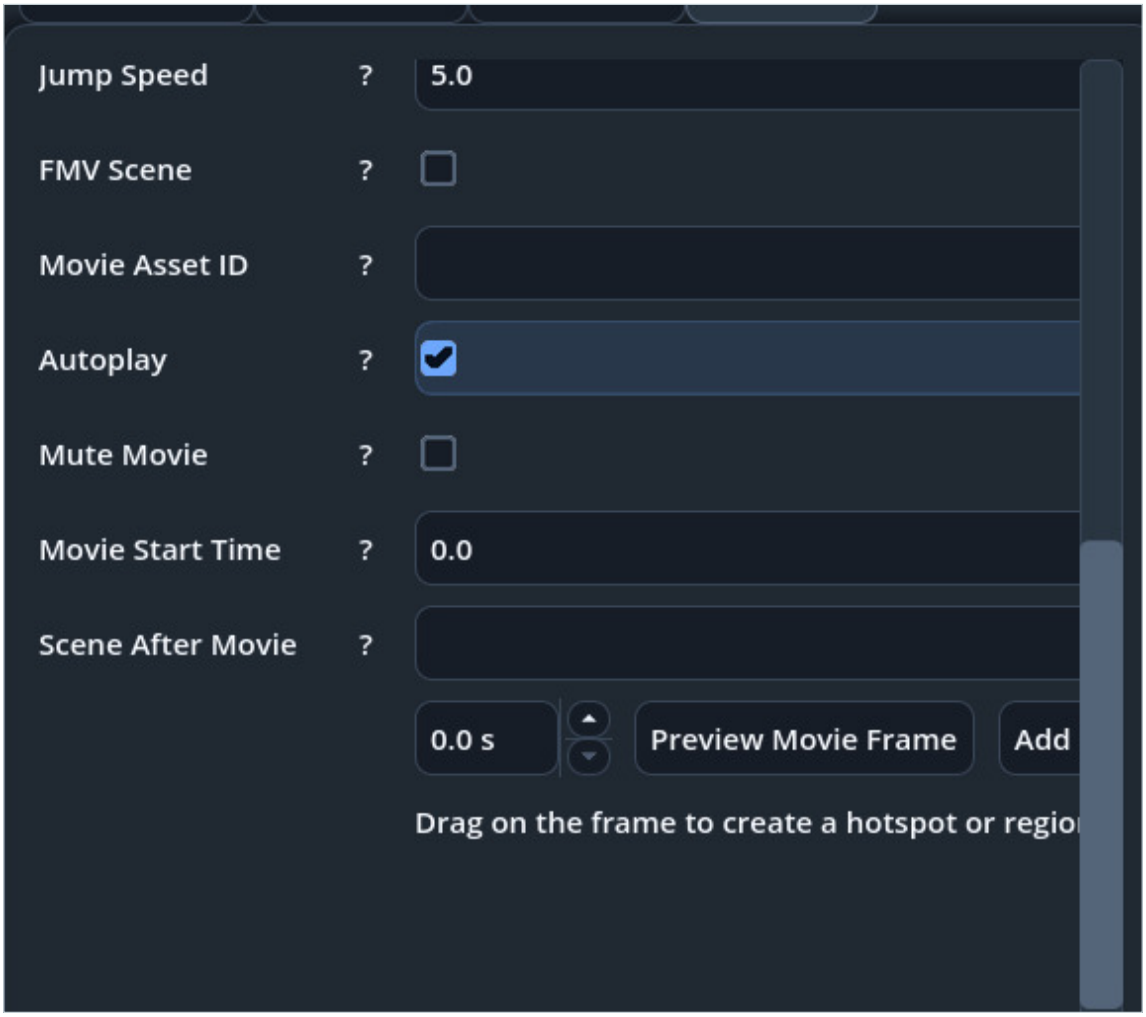


Figure 57. FMV settings are available in the room Inspector. Assign an imported video asset before trying to preview its frames.

For a movie scene, import the video first, enter its exact asset ID, enable FMV Scene, set its starting time, and choose what follows. Decide whether the player character should be hidden and whether existing GUIs should remain visible. Test actual playback and sound; the movie panel alone is not a playback test.

The **Movie Frame Editor** lets you enter a time and click **Preview Movie Frame**. **Add Movie Stop** makes a timed stop marker. Drag over the displayed frame to create a hotspot, or enable **Region** before dragging to create a movie region.

Widen the Inspector or use its horizontal scrollbar if a movie-editor row extends beyond the panel. Select a marker to edit its start/end timing, optional destination scene, and Actions JSON. Click **Apply Marker**, then apply the room Inspector and save the project. **Remove Marker** removes the selected marker.

A movie hotspot is a timed rectangle on the movie frame, not a 3D event volume on the workshop floor. If the picture moves, choose a useful active time interval or create additional hotspots for the relevant moments. Test before, during, and after that interval.

The action catalog provides **Start FMV**, stop, pause, resume, and seek controls for scripted playback. Start/end limits and completion scripts belong to the action's own setup. A full movie-editing workflow is a separate topic; these are the Scene Designer controls that connect it to a room.

27. Drive scene features with actions or scripts

The Inspector is where you author the initial state. Actions and scripts are how that state changes while the player is playing.

Choose the event that should start the work: a room entry, a trigger, an object's verb, an inventory rule, or another supported event. Pick the action, fill in the target and parameters, then apply. Use **Add Next Action** for a sequence. Use a visual script when the sequence needs conditions, loops, or more explicit flow.

Job	Useful action identifiers
Activate and blend shots	`ActivateCamera`, `ChangeCamera`
Change advanced follow settings	`SetCameraBehavior`, `SetCameraTargets`, `SetCameraPriority`
Orbit or shake	`CameraOrbit`, `StopCameraOrbit`, `CameraShake`, `ShakeScreen`, `StopScreenShake`
Change camera limits	`SetRoomCameraBounds`
Change the room	`ChangeScene`
Show/hide or enable/disable objects	`SetObjectVisible`, `SetObjectEnabled`, `SetTriggerEnabled`
Move or place characters	`CharacterWalkTo`, `CharacterTeleport`, `MoveNPCToRoom`, `AddCharacterToScene`
Waypoints and following	`AddCharacterWaypoint`, `CharacterWalkWaypoints`, `CharacterWalkPath`, `ClearCharacterWaypoints`, `FollowCharacter`, `StopFollowCharacter`
Face something	`FaceCharacter`, `FaceObject`, `FaceCoordinates`
Object motion and glow	`SetMotion`, `ToggleMotion`, `SetGlow`, `ToggleGlow`, `ToggleRGBString`
Change room appearance	`SetSceneBrightness`, `SetSceneTint`, `SetRetroStyle`, `SetSnowAccumulation`, `MeltSnow`

Job	Useful action identifiers
Temporary screen effects	`ZoomBlur`, `StopZoomBlur`, `FadeScreen`, `ScreenTransition`
Attach props or particles	`AttachObject`, `DetachObject`, `AttachParticleEffect`, `DetachParticleEffect`, `AttachPoints`, `AlignAttachPoints`
Spawn or change objects	`SpawnObject`, `DespawnObject`, `ChangeObjectProperty`, `AddAssetToScene`, `CreatePrimitive`, `CreateProcedural`
Stage changes	`SetRoomDimension`, `Set2DCamera`, `Set2DLayerVisible`, `Set2DEntityDepth`, `Set2DParallax`
Sound and music	Play, stop, pause, resume, and fade actions for sound and music.
Hand work to another tool	`RunVisualScript`, `CallScriptFunction`, `StartDialogue`, `StartCutscene`

The table uses internal action identifiers so the names remain unambiguous. Some displayed menu labels are friendlier: `ChangeScene` appears as **Change Room** in native event forms, and `ActivateCamera` appears as **Activate / Blend to Camera**.

A small practical sequence

For an interactive display cube, set its initial motion in the Inspector. Give its Use verb a **SetMotion** action targeting the cube with Enabled off. Add a message confirming it has stopped. Apply and test. Once that works, use a Boolean variable and a visual-script condition to switch between on and off.

This avoids the common mistake of changing a camera or object record in the editor and expecting it to be a timed event in the game. The authored value is a starting state; the action supplies the moment of change.

The text scripting languages share the common action catalog through their runtime helpers. Use the action's actual parameter fields and IDs, rather than guessing a property from its visible English label. The scripting editor's definitions and dedicated language documentation explain the entry function and syntax for each language.

SetRoomDimension is not permission to bypass a room's permanent family rules. Keep stage variant changes within the supported stage family. For a genuinely different room family, author a separate room and change scenes.

Wait only where waiting is intended

An action that starts an animation or motion does not necessarily wait for that animation to finish. Camera activation has an explicit **Wait for Blend** option. Use the action's timing controls or a visual-script wait where the next step depends on completion.

Do not disable player input and forget the matching step that enables it again. For a cutscene, use the cutscene system's restoration options where appropriate, then test skipping, interruption, and the normal ending.

28. Check the scene before calling it finished

Save a clean copy before the final pass. Use **Validate**, fix broken references, then test the scene through its normal entry route. A scene that works only when launched directly can still have a broken doorway or entry sequence.

Test	Look for
First visit and return visit	First-time events run once; ordinary entry events run on each visit.
Walk every edge and narrow passage	Correct navigation, clearance, collider size, and approach distance.
Stop and reverse direction	Follow camera settles, stays within limits, and does not reveal blank space.
All useful verbs and inventory items	Correct target rule, condition, and fallback response.
Pointer enter, press, release, and leave	Hover/click feedback and the correct normal state afterward.
Room changes in both directions	Correct destination, spawn, facing, and player transfer.
Sound near/far and across room changes	Correct volume, looping, fade, and cleanup.
Stage scroll and baseline crossings	Aligned paths, stable layer coverage, correct occlusion and feet.
Save, reload, and continue	Puzzle state, first-time events, and restored scene behavior.
A representative distributable build	Renderer, media files, performance, and input behavior match the target machine.

Troubleshooting by symptom

Symptom	First things to check
The character stops at an invisible line	Navigation edge, blocked area, physical collider, and the character's radius. Grid spacing is unrelated.
The character stops short of a door	Custom collider dimensions and approach distance. Event volume alone is not the physical obstacle.
No reflection of the character	Live Mirror versus Room Capture; reflection strength, axis, opacity, tint alpha, and overlapping walls.
A selected effect does nothing	Apply Changes, the owning record, enabled override, actual renderer, and whether an imported shader overrides the layers.
No visible depth-of-field blur	Actual preview backend and transparent-layer restrictions. Compatibility does not support native DOF here.
Following does nothing	Active camera, correct target ID, enabled behavior, inheritance, and an actual running Test Scene.
A group shot crops one character	Maximum distance, framing padding, FOV, room bounds, and whether both targets exist in the current room.
A parallax layer is invisible	Opaque artwork in front of it, layer ordering, image coverage, and whether the camera actually moves.
The whole character vanishes behind scenery	Hide Entire Character at Baseline and the baseline's position/side.
A sprite floats or looks tiny	Transparent padding, visible-height fitting, foot anchor, source frame grid, and scale.
A new room fails validation	Duplicate reusable character placements, duplicate IDs, missing assets, and destination references.
An event repeats too often	Entry versus first-entry, hover enter versus click, and whether a trigger needs to disable itself.
The preview gets slow	Visible live mirrors, real lights on RGB strings, shadows, particle count, shader stacks, and render quality.

This build can report an **unknown ambient-sound entity type** warning for an ambient sound created by its own scene tools. Check the record and file reference, and test its runtime behavior; do not rename its type to a different object merely to silence the warning. Treat genuine errors separately. During this guide's checks, a duplicate character placement was a real blocking error and was corrected before the running room tests.

A sensible performance pass

Keep one known-good shot and compare a single change at a time. Turn off an expensive effect temporarily, test, and restore it if it is worth the cost. Reduce unnecessary shadow-casting lights, particle density, or visible mirrors before reducing the scene's readability.

The game renderer and the currently running preview renderer may differ. Read the preview diagnostic. Restart PCA when changing its backend; restarting only the preview does not rebuild the editor around a different renderer.

Performance, Balanced, and Authored Quality are quality choices separate from renderer selection. A cheaper quality setting does not mean deleting the adventure's actions, scenes, or puzzles. Still test the final look and input on the machine you plan to support.

> **Before you continue**
> Save, reopen, validate, enter the room normally, complete the interaction, leave, and return. Then repeat the important checks in a distributable game build.

29. Where every Scene Designer tool is covered

Use this index with the first two Scene Designer guides. “Part 3” means this guide. It covers the remaining Scene Designer feature families exposed by the 0.86.3.4 build; it is not an assertion that every slider combination or third-party shader has been tested on every graphics backend.

Tool or feature	Where to read
Create/save a room; room family choice	Part 1, Chapters 1–3; Part 3, Chapters 1 and 20.
Scene Objects, Inspector, Apply versus Save	Part 1, Chapters 2–3; Part 3, Chapters 1–2.
Move, rotate, scale, axes, camera navigation	Part 1, Chapter 3; Part 3, Chapter 2.
3D, Designer, Mesh, stage views, view presets	Part 3, Chapter 2.
Frame scene/selection; grid/snap	Part 1, Chapter 3; Part 3, Chapters 2–3.
Search, type filter, names/IDs, folders, lock, duplicate/copy/paste	Part 1, Chapter 22; Part 3, Chapter 2.
Editor Visibility and Clean 2D Room	Part 3, Chapters 2 and 24.
Primitives, floor, walls, color and texture	Part 1, Chapters 4–8.
Procedural furniture and scenery, additional street generators	Part 1, Chapter 12; Part 3, Chapter 25.
Import/drop actual 3D assets; scale and origin	Part 2, Chapters 2–6.
Blocking, custom box collider, event volume, approach	Part 1, Chapter 11; Part 2, Chapters 5 and 15; Part 3, Chapters 3 and 28.
Navigation rectangles, areas, polygons, vertices, winding, blockers	Part 1, Chapter 10; Part 3, Chapter 3.
Surface walking, gravity, jump permission	Part 3, Chapter 3.
Characters, player choice, models and animation states	Part 1, Chapters 13–16; Part 2, Chapters 7–10.

Room Inspector pages

Page	Where to read
General: identity, room audio, background, player visibility/view, GUI hiding, display names, player zoom/pan override	Part 3, Chapters 1–4 and 13; see the notes below.
2D / 2.5D Room: family, background, background animation, dimensions and ground	Part 3, Chapter 20.
2D Camera: framing, view height, roll, follow, limits	Part 3, Chapter 21.
2D Navigation: shared-navigation explanation	Part 3, Chapters 3 and 21.
2D Layers & Objects: parallax and walk-behind records	Part 3, Chapters 22–24.
Screen Overlay	Part 3, Chapter 18.
Camera & View: viewport, default camera, bounds, dynamic override, default-camera DOF	Part 1, Chapters 16–19; Part 3, Chapters 4–7 and 18.
Lighting & Fog	Part 2, Chapter 20; Part 3, Chapters 8–10.
Post Process: cinematic, stack, weather, outlines	Part 3, Chapters 16–17.
Room Events	Part 3, Chapter 19.
Transitions, additional gravity and FMV controls	Part 3, Chapters 3, 19, and 26.
Custom Properties and Raw JSON (Advanced)	Part 3, Chapter 26.

Hide Player Character suits a first-person or visual-novel view. **Player View Override** can inherit or choose the room's supported first-/third-person treatment. **Hide All GUIs on Room Load** controls the initial GUI state; use actions to show the required interface afterward. **Object Display Names on Hover** can inherit, always show, or never show names in the room. Check these settings deliberately when making an unusual room, rather than using them to hide an unrelated setup problem.

Other Scene Inspector pages

Record	Main pages and coverage
Camera	General; Position & View; Dynamic Follow; Advanced Camera; Depth of Field; Custom Properties; Raw JSON. Part 1, Chapters 16–19; Part 3, Chapters 4–7 and 26.
Ordinary object	General; Object Events; Verb Actions; Inventory Actions; Visual Adjustments; Visual Effects; RGB Lighting; Pointer Feedback; 2D / 2.5D; Custom Properties; Raw JSON. Parts 1–2 for core placement/actions; Part 3, Chapters 9–14, 23, and 26 for the remaining tools.
Placed character	General; Character Settings; Player Settings; Character Events; Verb Actions; Inventory Actions; Visual Adjustments; Visual Effects; Pointer Feedback; 2D / 2.5D; Custom Properties; Raw JSON. Part 2 for model/state work; Part 3 for the shared appearance, interaction, attachment, and stage tools.
Light	General; Light Settings; Lens Flare; God Rays; Visual Adjustments; 2D / 2.5D; Custom Properties; Raw JSON. Part 3, Chapters 8–9, 23–24, and 26.
Particle	General; Particle Settings; Motion & Emission; Particle Events; Visual Adjustments; 2D / 2.5D; Custom Properties; Raw JSON. Part 2, Chapter 22; Part 3, Chapters 14 and 24.
Trigger	General; Trigger Events; Verb Actions; Visual Adjustments; Custom Properties; Raw JSON. Part 2, Chapter 19; Part 3, Chapters 3, 12, and 27–28.
Region	General; Region Events; Verb Actions; Visual Effects; Custom Properties; Raw JSON. Part 2, Chapter 21; Part 3, Chapter 15.
Navigation Area	General; Shape; Custom Properties; Raw JSON. Part 3, Chapters 3 and 21.
Blocked Area	Bounds, identity/state, custom properties, and Raw JSON. Part 3, Chapter 3.
Spawn	Identity/state, position, facing, custom properties, and Raw JSON. Part 3, Chapter 19.
Ambient Sound	Identity/state, proximity audio, transform, custom properties, and Raw JSON. Part 3, Chapter 13.
Stage Background / Parallax / Walk-Behind / Camera	Their dedicated record forms. Part 3, Chapters 20–23.

A reusable character definition deliberately has fewer scene-appearance pages than a placed character. Select the placed instance when changing room-specific appearance or pointer feedback. Native tab strips can scroll sideways; use their arrows when a tab is out of sight.

Screen-preset index

Family	Presets
Retro	PSX, N64, CRT, VHS-CRT-2, classic dithering, pixelized.
Color and grading	Black-white, sepia, posterized, invert, warm, cool, vignette, film-grain, photography.
Drawn and stylized	Cel-shaded, toon, toon-outline, low-poly, color-sketch, color-crayon, hand-drawn, metallic-toon, stencil-drawing.
Distortion and fantasy	Distorted, aurora, frostbite, broken-glass, wind-sway, glass-pixel, hologram, screen-flicker, screen-fire, heat-haze.
Light and weather	Bloom, horror-fog, snow, rain, underwater.
Off	None.

Some presets are also offered on object surfaces. The owner and shader type determine what receives the effect. A preset's presence in a menu does not make a surface shader and a screen shader interchangeable.

What remains in other designers

Asset importing, reusable character authoring, full animation-state editing, GUIs, dialogues, cutscene timelines, text parsers, translations, build tools, and the full scripting languages have their own designers. This series covers their Scene Designer connections rather than replacing each separate manual.

The Advanced Camera integration uses Phantom Camera by Marcus Skov (MIT License). Its upstream developer credits and license are retained with the engine's bundled component. The artwork shown here is the engine's bundled workshop and storybook content. All editor pictures in this guide are captures of the running application; close-ups are cropped for readability.