PCA 3D ADVENTURE STUDIO

Shaders, Effects and Layered Scenery

Particles, attach points, parallax and screen overlays.



Figure 1. A live hand effect in the Effects Workshop. The particle follows a bone-bound attach point.

A practical, illustrated guide to building a look, placing effects in the scene and controlling them during play.

For PCA 3D Adventure Studio 0.86.3.4

How to use this guide

Start with a plain working scene. Add one effect, test it through the gameplay camera, then decide when it starts and stops. That approach makes the controls much easier to understand.

Lessons 1-8 explain where effects belong and how to use the shader stack. Lessons 9-23 cover room, object, region and GUI effects, including retro styles, fog, mirrors, heat and imported shaders.

Lessons 24-38 cover every standard Particle preset, custom images, motion, timing, stopping and performance. Lessons 39-49 cover point placement, animated bones, offsets, local coordinates and attachment actions.

Lessons 50-58 build and test scrolling image layers. Lessons 59-65 cover the separate Screen Overlay system. Lessons 66-78 connect everything to events, visual nodes and text scripts, then cover cleanup and final testing.

Screenshots and example records

All editor and gameplay illustrations are actual native captures. The small workshop uses bundled Clockmaker character models and existing engine artwork. The stage example uses a painted scene and a transparent foreground image.

The instructions are platform-neutral. Theme, UI scale, display width and rendering backend can change the visible layout. Use the named controls and record IDs rather than relying on exact screen coordinates.

No separate practice files are included. Build the examples in a project of your own, using your own IDs where appropriate. Save a copy before experimenting in an existing game.

A useful distinction

A saved property is not always an immediate live redraw. This guide distinguishes fields that refresh the current effect from configuration that needs a fresh room presentation. It also distinguishes a visual effect from gameplay logic, light, sound and collision.

Code

JSON needs ordinary double quotes, commas and matching brackets. GDScript indentation matters. PCA Script examples preserve the physical expression lines required by its parser. The code belongs in project script records; no engine source changes are needed.

Contents

1. Decide where the effect belongs . . . 5
2. A plain-English rendering vocabulary . . . 6
3. Find the controls without losing the room . . . 7
4. Prepare a useful effects workshop . . . 8
5. Apply, save and test are different steps . . . 9
6. Global shaders and room overrides . . . 10
7. Build your first room shader stack . . . 11
8. Understand layer order and shared values . . . 12
9. Color treatments: start with a light touch . . . 13
10. Toon, sketch, crayon and outlines . . . 14
11. Analog, animated and special presets . . . 15
12. Screen weather and snow accumulation . . . 16
13. Give a room a PSX-style treatment . . . 17
14. Use the N64 filter deliberately . . . 18
15. Set up horror fog with a clear foreground . . . 19
16. Grade the scene without piling on presets . . . 20
17. Put a shader on one object . . . 21
18. Make a glowing object light its surroundings . . . 22
19. Reflections: room capture versus a live mirror . . . 23
20. Heat haze: screen, surface or particles? . . . 24
21. Tint or blur a region of the scene . . . 25
22. Import a shader without guessing its type . . . 26
23. GUI shaders and world effects stay separate . . . 27
24. Add your first particle emitter . . . 28
25. Choose a particle preset for the job . . . 29
26. Amount, lifetime and one-shot timing . . . 30
27. Shape particle motion . . . 31
28. Size the emission area and artwork separately . . . 32
29. Give particles a custom image . . . 33
30. Build a brazier from two effects . . . 34
31. Make rain, snow and hail cover the room . . . 35
32. Clouds, floor mist and a small swarm . . . 36
33. Author a lightning bolt . . . 37
34. Put a lightning ball above a hand . . . 38
35. Trigger an explosion more than once . . . 39
36. Stop emission, hide an effect or detach it . . . 40
37. Which particle changes update during play? . . . 41
38. Keep particles affordable and readable . . . 42
39. Attach points are named sockets . . . 43
40. Place a point in Asset Designer . . . 44
41. Bind the point to the moving bone . . . 45
42. Use per-animation point adjustments sparingly . . . 46
43. Know which record owns the point . . . 47
44. Attach the authored particle in the Inspector . . . 48
45. Fix position and direction after attaching . . . 49
46. Local coordinates decide what old particles do . . . 50
47. Attach and detach through an action list . . . 51
48. Carry a prop and attach an effect to it . . . 52
49. Point contacts and attachment are separate . . . 53
50. What parallax actually needs . . . 54
51. Prepare a scrolling 2D Stage . . . 55
52. Set the camera’s scroll range . . . 56
53. Add and select a Parallax Plane . . . 57
54. Choose sensible parallax factors . . . 58
55. Give each layer enough spare image . . . 59
56. Parallax is not a walk-behind mask . . . 60
57. Current stages and older native 2D rooms . . . 61
58. Test the parallax as a moving composition . . . 62

59. Screen overlays are decorative image layers . . . 63
60. Prepare artwork for a screen frame . . . 64
61. Set a default overlay for the game . . . 65
62. Give a room its own overlay . . . 66
63. Choose how the overlay is shaded . . . 67
64. Add camera-driven overlay parallax . . . 68
65. Overlay size, transitions and live changes . . . 69
66. Put the effect on an event . . . 70
67. Build the same flow with visual nodes . . . 71
68. A complete script to enable the hand effect . . . 72
69. A complete script to stop and release it . . . 73
70. Switch the inherited screen look by script . . . 74
71. Change a room stack or a caller-owned effect . . . 75
72. Use the same actions in PCA Script . . . 76
73. Combine a burst, sound and zoom blur . . . 77
74. Add shake, snow changes and restrained motion . . . 78
75. Make cutscene cleanup explicit . . . 79
76. Diagnose a missing or misplaced effect . . . 80
77. Diagnose parallax and overlay problems . . . 81
78. Finish and test the complete scene . . . 82
Appendix A. Particle property reference . . . 83
Appendix B. A complete screen-style value . . . 85
Appendix C. A current-stage parallax plane . . . 86
Appendix D. Overlay configuration reference . . . 87
Appendix E. Renderer and setup reference . . . 88
Appendix F. Credits for shader work . . . 89
Version and verification notes . . . 91

1. Decide where the effect belongs

A shader changes how something is drawn. A particle emitter produces a collection of moving pieces. An overlay puts artwork over the view. Parallax makes different layers appear to move at different rates. They can work together, but they solve different jobs.

You want to change…	Start here
The appearance of a whole room	Global Settings → General (Default Global Scene Shader Stack), or the room Inspector → Post Process.
One model, sprite or primitive	Select that scene object → Visual Effects → Shader Layers and Effects.
A small area of the view or a floor patch	Select a Region → Visual Effects.
An interface window or control	GUI Designer → GUI Window or Element Shader.
Fire, rain, sparks or drifting puffs	Scene Designer → Add to Scene → Particle.
An effect that follows a hand or machine	Create the particle, create the target attach point, then link them.
Painted scenery with depth	A 2D Stage’s background, Parallax Planes and scrolling camera.
A decorative frame over the game	Global Settings → Screen Overlay, with optional room overrides.

Choose the smallest surface that does the job. A heat plume over a brazier belongs near the brazier. A dream sequence might need a room-wide treatment. A glowing inventory icon belongs to the GUI.

> An effect that looks like fire does not automatically add light, sound or damage. An image of glass is not automatically a mirror. Build those behaviors separately when the scene needs them.

2. A plain-English rendering vocabulary

Term	Meaning in this guide
Material	The surface description: its image, base color, roughness, metallic response and other drawing rules.
Shader	A small graphics program that decides how a surface or screen image looks. Built-in presets let you use one without writing it.
Screen / post process	A treatment of an already-rendered room image, such as sepia, grain or pixelation.
Layer / slot	One entry in a shader stack, with its own preset and intensity. A stack has up to eight entries.
Emitter	The source of particles. Its position is where new particles are created.
Attach point	A named position and orientation on an object or animated bone. It is often called a socket in other engines.
Local coordinates	Coordinates relative to the emitter or object, rather than fixed in the room.
Alpha / opacity	Transparency. A transparent center in a PNG lets the room show through.
Parallax factor	How much an image layer appears to travel as the camera moves.
World units	The room’s X, Y and Z measurements. In a standard 3D room Y is height; in a current 2D Stage the image lies across X and Z.

You do not need to learn shader programming to follow the main tutorials. Use the preset menus and the named fields. The later scripting pages show how to change those same authored settings during play.

> “Glow,” “Bloom,” “Emission” and “Light” describe related appearances, but they are separate controls. A bright-looking particle can be visible without casting any light on the floor.

3. Find the controls without losing the room

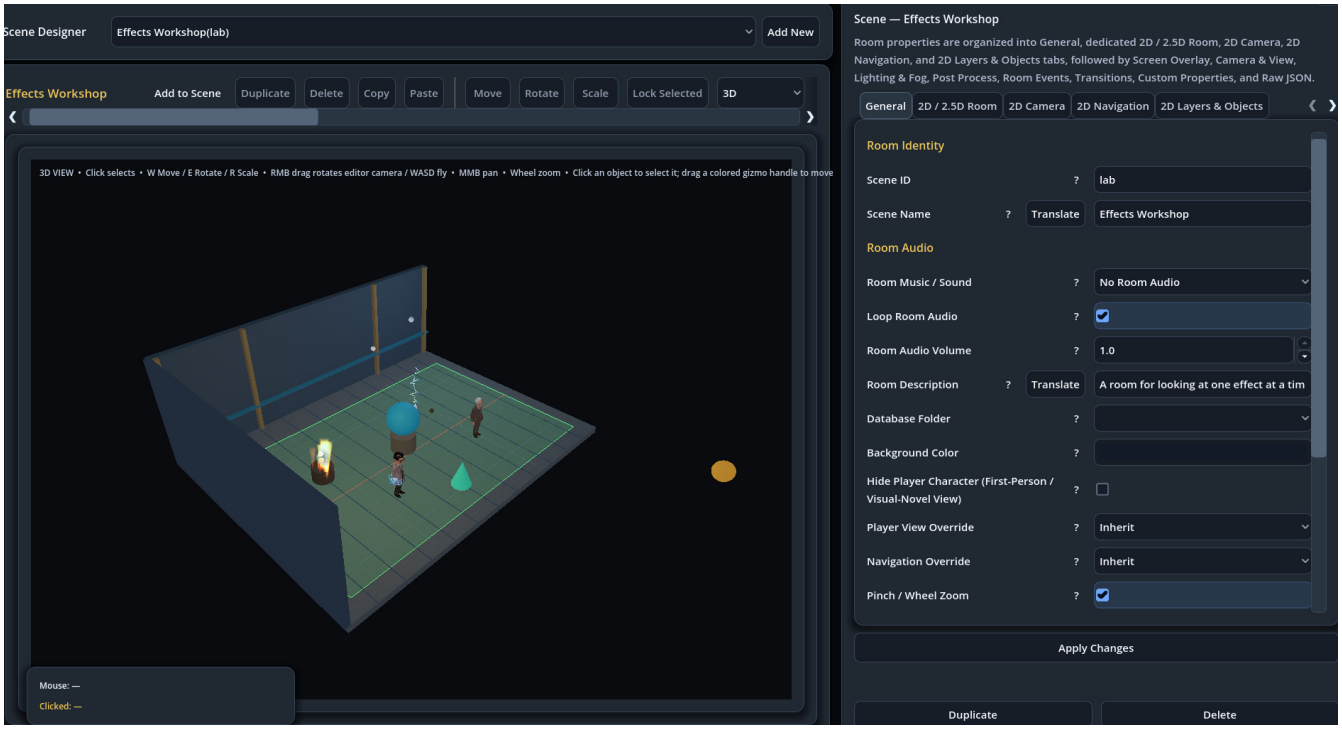

Figure 2. The Effects Workshop in Scene Designer. The room and its objects remain ordinary editable scene records.

1. Open your project and choose Scene Designer. Select the room you want to work on.
2. Show the Scene Objects panel and Inspector from Panels if either is hidden. The Scene Objects list is the most dependable way to select a small or transparent effect.
3. Select the room itself for its Screen Overlay, Lighting & Fog and Post Process tabs. Select an object for its own Visual Effects or Particle Settings tabs.
4. Use the arrows beside the Inspector tabs to reveal tabs that do not fit. Scroll inside the active tab to reach its lower controls.
5. Keep the gameplay camera in mind. An effect framed in the editing view may be behind the camera or too small in the actual game.

The illustrations use a clean desktop layout. Your theme, UI size and screen width can change the position of a button. The record names and tab labels identify the controls more reliably than their pixel positions.

> The main Inspector and Scene Designer’s own room-property controls can expose the same record. Apply changes in the editor you are using, then save the project. Use Test Scene to judge the actual game view.

4. Prepare a useful effects workshop

Use a small room before adding effects to an elaborate game. A floor, a back wall, one colored object and one animated character give you enough detail to judge what changes. Keep a few lines or textures in the background: distortion is much harder to see against a flat wall.

Example record	Purpose
Effects Workshop (lab)	A 12-by-9 world-unit room with a fixed wide camera.
Shader Orb (shader_orb)	A cyan sphere on a pedestal for trying surface presets.
Brazier (brazier)	A small prop with a named flame_top point.
Brazier Flame (flame)	An authored Fire particle emitter.
Brazier Heat (heat_plume)	A separate Heat Haze emitter.
Professor Ivo (ada)	The live player instance, with a bone-bound right_hand_grip point.
Hand Lightning Ball (hand_orb)	A small Lightning Ball, initially not emitting.

These IDs belong to the examples. Use your own names and IDs in your project. A display name makes a record easy to find; the ID is what an action or script uses. A selector may show Professor Ivo(ada), while the action stores ada.

1. Save a copy of a project you can safely experiment in.
2. Confirm that the plain room renders correctly before enabling a screen shader.
3. Add one effect at a time and make a note of the value that looks good.
4. Test both a close shot and a wide shot. Particle size, line width and grain can look very different at different distances.

> No practice project is supplied with this guide. All examples can be built in your own project. The captured room uses the bundled Clockmaker character assets and existing engine artwork.

5. Apply, save and test are different steps

Step	What it proves
Change a field	The form contains the value you want. Some previews may update while editing.
Apply Changes	The edited record is written into the open project.
Save Project	The project on disk includes the applied record.
Test Scene	The room’s real camera, drawing and runtime behavior can be checked.
Play / Restart	Startup presentation, room-entry actions and the complete game route are exercised.
Build Game and run the output	The assets and renderer settings are present in the actual distributable.

For a shader, test its appearance after applying it. For a one-shot particle, restart the test so you see its beginning. For a moving attachment, play an animation or make the character walk. For parallax, move the camera; a still image cannot show scrolling.

When a setting appears to do nothing, check which record owns it. A room using its own Custom stack will continue to override the global stack. A selected imported shader replaces built-in effect layers on that surface. An overlay with No Shaders deliberately stays outside the room treatment.

1. Change a single obvious value, such as Sepia at intensity 1.
2. Apply, save and restart Test Scene.
3. If it works, return to the intended subtle value.
4. If it does not, check the source/override selector before changing ten unrelated sliders.

> Keep a clean baseline. Turning off an effect should be a deliberate setting or action, not an attempt to remember every slider’s original value.

6. Global shaders and room overrides

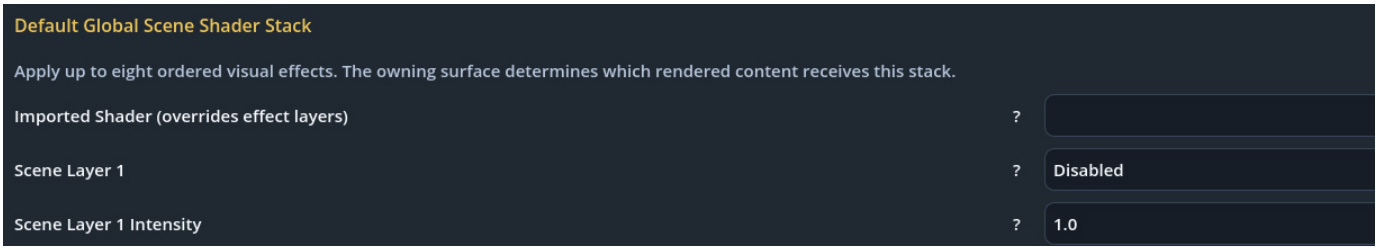

Figure 3. The default shader stack is in Global Settings → General. Rooms decide whether to inherit it.

Room: Scene Shader Source	Result
Use Global Scene Shaders	Uses Global Settings → General (Default Global Scene Shader Stack). A later global change affects this room.
Use This Scene’s Shader Stack	Uses the room’s saved custom stack. It replaces the inherited stack rather than adding another eight layers.
Disable Scene Shaders in This Scene	The room opts out of the screen stack. This does not remove its object materials, particles or separate GUI effects.

For the game-wide look, open Global Settings → General and scroll to Default Global Scene Shader Stack. It is a section in General in this build, rather than a separate tab. A memory scene, underwater room or special vision mode may need a Custom room treatment. Use Disabled for a room that must remain clear.

The room Inspector puts this selector under Post Process, beneath its cinematic controls. Scene Designer’s room form also calls the section Room Screen Shader Stack / Post Process. They refer to the room’s same saved post-process data.

> The Cinematic Override selector and Scene Shader Source selector control different groups. Setting the shader stack to Disabled does not automatically disable a separate cinematic grade.

7. Build your first room shader stack

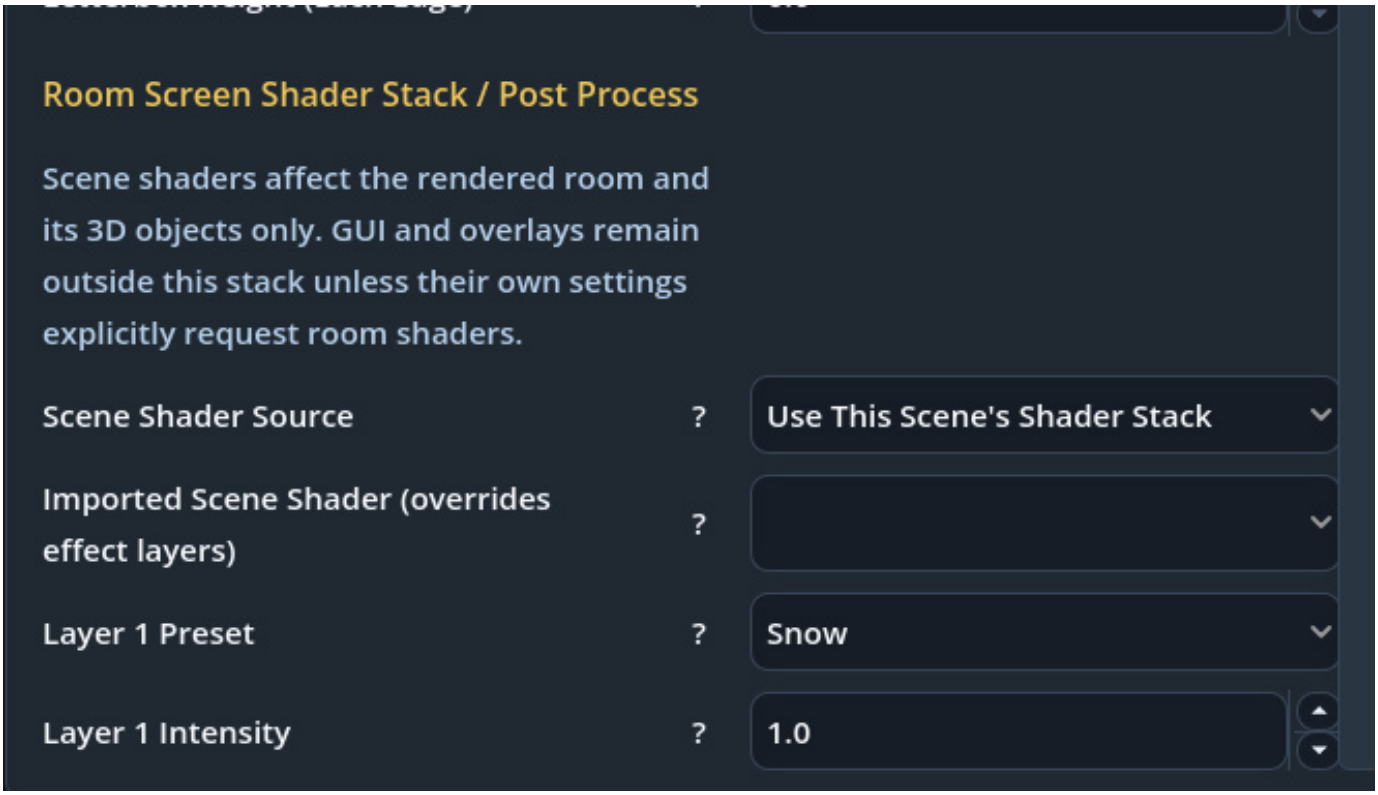

Figure 4. The room’s Post Process tab contains the source selector and numbered shader slots.

1. Select the room and open Post Process.
2. Set Scene Shader Source to Use This Scene’s Shader Stack.
3. Leave Imported Scene Shader empty. An imported scene shader overrides the preset layers.
4. Choose Sepia in Layer 1 Preset. Set Layer 1 Intensity to 0.6.
5. Choose Vignette in Layer 2 Preset. Start with intensity 0.2. Leave the other slots at None.
6. Apply Changes, save and use Test Scene. The room should look warmer, with slightly darker edges. The gameplay HUD should remain readable.
7. Return Layer 1 to None to compare. Restore it when you are satisfied.

> The same numbered slots are used for the other presets. Dedicated style controls appear farther down the form when the relevant preset is selected.

8. Understand layer order and shared values

Each layer has a preset and its own intensity. Most of the fine controls belong to the whole stack. Two layers using Pixel Size do not receive independent Pixel Size sliders. Changing the shared value can change both appearances.

Control	How to use it
Layer Preset = None	Leaves that slot unused.
Layer Intensity = 0	Removes that layer’s visible contribution while retaining its selected preset.
Layer Intensity near 1	A good starting point for a full effect. A large value is not automatically better.
Pixel / Grain / Stroke Size	A shared scale for pixelation, grain or marks, depending on the selected preset.
Distortion Amount / Animation Speed	The strength and speed of relevant moving distortions.
Color / Toon / Art Levels	The number of broad color bands for stylized effects.
Edge / Ink Strength	The strength of relevant edge-detection or drawn-line effects.
Tint Color / Tint Strength	A final color wash; white with zero strength is a useful neutral setting.

The stack is ordered, but it is not eight independent camera renders. UV changes, such as pixelation and distortion, are combined before the image is sampled. Color treatments then run through the ordered slots. Swapping two color treatments may change the result; swapping any two presets does not promise a physically separate pass.

The exclusion checkboxes preserve light, shadow or RGB-lighting characteristics where the built-in effect supports that treatment. They do not turn lights off, remove shadows from the room or make a separate camera layer.

> When clearing a stack through code, clear the legacy preset too. Use {"layers":[],"preset":"none"}. An empty layers array alone can fall back to a retained older single-preset value.

9. Color treatments: start with a light touch

Preset	Useful starting point
Black and White	A flashback, security view or a deliberately monochrome room. Check that important objects still separate by brightness.
Sepia	An old photograph or memory. Try intensity 0.4-0.7 before committing to the full wash.
Warm / Cool	A simple mood shift. They color the image; they do not reposition a light or change its direction.
Posterized	A limited set of broad color bands. Adjust Color Levels before raising intensity.
Invert	An unusual vision or magical pulse. Reserve a strong inversion for a clear purpose.
Vignette	Darkens the outer view to favor the center. Keep exits and edge hotspots visible.
Photography	A camera-like treatment within the shared preset system. Judge it against the room’s existing lighting and grade.

1. Begin with one color preset, no extra tint and a neutral cinematic grade.
2. Use a close-up and a wide shot. Check faces, subtitles, exits and inventory objects.
3. Add a modest Vignette only after the main colors work.
4. If a puzzle relies on colors, test it under the final treatment. A blue wire and a green wire can become indistinguishable after heavy grading.

A color effect should support the scene. It should not force the player to hunt for a hotspot that was obvious before the effect was added. If you want an object to remain exceptional, give that object a readable shape, lighting or interface cue as well.

10. Toon, sketch, crayon and outlines

Preset / feature	What to adjust
Cel Shaded / Toon	Color Levels for broad bands, then intensity. Screen versions stylize the room image; they do not redraw the mesh topology.
Toon Outline	Color Levels and Edge / Ink Strength. It combines simplified colors with detected contours.
Color Sketch	Pixel Size for grain/stroke scale; Levels and Edge Strength for the drawn appearance.
Color Crayon	Pixel Size, Levels and Edge Strength. Expect saturated bands and irregular wax-like marks.
Hand Drawn	A drawn/squiggled image treatment. Check fine details and moving edges.
Classic Dithering	A patterned reduction of tones. Test at the game’s intended output size.
Low Poly	A stylized/faceted drawing treatment. It does not reduce your imported model’s polygon count.
Whole-Scene Silhouette Outline	A separate room option with color, thickness and Edges Only. Do not confuse it with a screen Toon Outline preset.

To try the whole-scene option, select the room, open Post Process and scroll to Whole-Scene Silhouette Outline. Enable Outline Everything in This Room. Choose a contrasting color and start with a small thickness. Enable Edges Only for contour treatment instead of the filled-surface appearance.

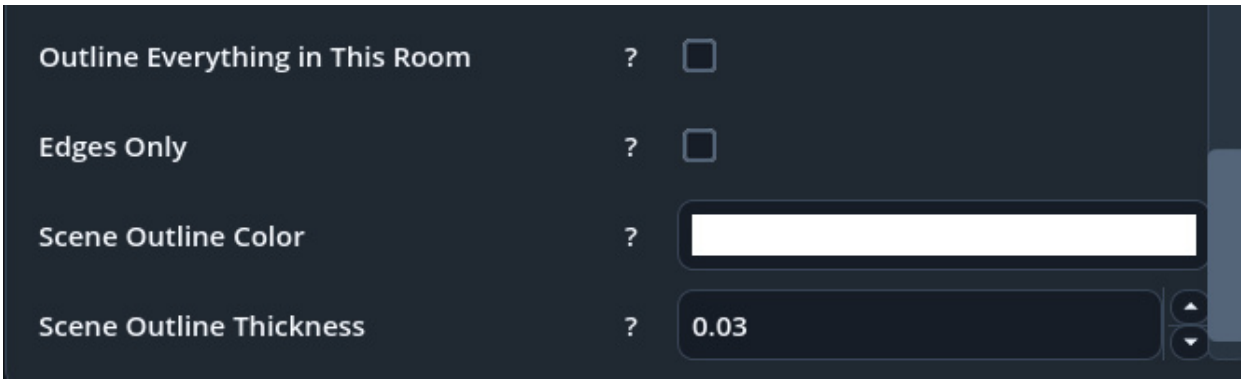

Figure 5. The room’s silhouette controls are separate from the eight post-process slots.

> Edges Only outlines eligible objects. It is not a mesh wireframe display, and it does not reveal every internal modeling edge.

11. Analog, animated and special presets

Family	Presets and typical use
Old displays	CRT and VHS / CRT 2 for scanlines and analog instability; Film Grain for texture; Screen Flicker for an unstable signal.
Moving backgrounds	Aurora for a stylized animated light pattern; Wind Sway and Distorted for movement/distortion. A room-wide sway moves the image, not the physical trees.
Ice and glass	Frostbite, Broken Glass and Glass Pixel for screen/surface treatments. They do not create breakable collision fragments.
Fantasy or machines	Hologram for a projected look; Screen Fire for drawn flame-like image treatment. A real world-space flame is a Particle object.
Metal and brightness	Metallic Toon and Bloom for their stylized surface/image response. A physical-looking metal or live mirror uses additional material settings.
Pixels and hot air	Pixelized changes sampling scale; Heat Haze distorts the background. A localized heat surface or emitter is usually better than treating the entire room.

Try these with one slot active. Start Animation Speed around 1, then make small changes to Distortion Amount and Pixel Size. The names of shared controls describe a toolkit, not a promise that every slider affects every preset.

1. For a television room, start with CRT at a moderate intensity.
2. Add a little Film Grain. Keep the text interface outside the room stack.
3. If you need a strong signal failure, trigger it briefly from an action or script and restore the ordinary stack afterward.
4. Test flickering and distortion with a reduced-effects option before using them for long scenes.

A preset is a drawing style. Broken Glass does not destroy a window; Hologram does not make an actor non-solid; Screen Fire does not burn nearby objects.

12. Screen weather and snow accumulation



Figure 6. Snow, rain, underwater and stencil controls live below the room’s shared stack settings.

Preset / controls	Use
Snow	Density, fall speed, angle, flake scale and color draw snowfall over the room image.
Snow Surface Accumulation	An additional treatment of eligible 3D surfaces. Tune amount, top-surface threshold, slope softness, noise and patchiness.
Accumulation Time	Seconds for the buildup to reach its authored amount. Zero avoids an animated buildup.
Rain	Density, speed, angle, streak scale and color control the screen rain.
Underwater	Tint, distortion, wave speed and caustic strength produce an underwater image treatment.
Stencil Drawing	Ink/paper colors, darkness threshold, edge strength and line width produce a high-contrast drawing.

Use screen weather when the whole view needs a consistent treatment. Use world-space particles when you need a source at a particular position, such as snow outside a window or sparks near a machine. Combining both can work, but it can also double the apparent density.

> Snowfall particles and a snow screen layer are separate systems. Particle snow does not automatically accumulate on meshes. Accumulation is a visual coating, not snow-depth collision or a change to navigation.

13. Give a room a PSX-style treatment



Figure 7. PSX has dedicated controls in the screen shader settings; shared Pixel Size is a different field.

1. Select the room → Post Process → Use This Scene’s Shader Stack.
2. Choose PSX in one slot and start with intensity 1. Leave the other slots at None while tuning.
3. Start PSX Pixel Size at 3 and PSX Color Bits per Channel at 5.
4. Set PSX Dither Strength near 1 for the ordered pattern, or lower it for a cleaner image.
5. Leave PSX Vertex Jitter at 0 initially. Raise it gently only if visible geometry wobble suits the game.
6. Apply, save and test camera movement as well as a still frame.

The screen palette is applied to the room image. Eligible ordinary 3D surfaces can also use the console geometry integration. An imported custom ShaderMaterial keeps its own geometry behavior, even though the completed object’s pixels can still receive the room screen treatment.

> This is a console-inspired visual style, not a PlayStation emulator. Do not expect every modern material feature to look the same under a deliberately simplified retro surface.

14. Use the N64 filter deliberately



Figure 8. The same workshop through the N64 room treatment. Keep the interface outside the room stack when clarity matters.

Control	Starting point
Layer preset / intensity	N64 / 1 for a clear comparison.
N64 Pixel Size	Try 2; raise it only until the scene has the intended scale.
N64 Three-Point Filter Strength	Try 1 for the full treatment, then compare with 0.5.
Other active retro styles	Disable them while tuning so you know which treatment you are seeing.

The N64 option combines the room’s image treatment with three-point texture sampling on eligible standard 3D surfaces. It does not need PSX jitter. Models using their own custom shaders retain that shader’s material behavior.

Use real in-game camera distances. A filter that looks appealing on a close wall may make a distant item icon disappear into the background. GUI portraits, dialogue and inventory remain separate unless you add their own effects.

> The older Retro Rendering controls and Set Retro Rendering Style action are still present for existing projects. For a consistent new screen look, start in Scene Shaders and avoid accidentally enabling both routes.

15. Set up horror fog with a clear foreground



Figure 9. Horror Fog in the workshop. Near surfaces remain more visible while the rear of the room fades.

1. Choose Horror Fog in the room’s Custom screen stack.
2. Pick a fog color that matches the scene’s darkness rather than starting with pure white.
3. Set Start Distance where the foreground should still be clear. For this small workshop, try 3.
4. Set End Distance farther into the room, for example 13. Keep it greater than Start Distance.
5. Adjust Horror Fog Amount before adding grain or a strong vignette.
6. Apply and test from the actual camera. Move a character toward and away from the camera to judge the depth.

In a 3D room the depth pass reads opaque scene depth. Transparent artwork does not provide its own solid depth surface. In a 2D room/stage, the treatment supplies its mist, grain and vignette appearance; it cannot infer the depth of objects painted into one flat image.

> Horror Fog is separate from the room’s normal Lighting & Fog settings. Combining both is possible, but start with one. Too many fog controls at once make it difficult to tell which system is hiding the scene.

16. Grade the scene without piling on presets



Figure 10. Cinematic Presentation has its own enable switch and room override.

Control	A practical interpretation
Exposure (EV)	Raises or lowers image brightness after lighting. It does not move a light.
Contrast / Saturation	Neutral values are 1. Start with small changes, such as 1.05 contrast or 0.9 saturation.
Color Warmth	Neutral is 0. Positive and negative changes shift the mood.
Vignette / Film Grain	Neutral is 0. Keep both subtle around interactive scenery.
Letterbox Height (Each Edge)	Fraction of the image covered at each edge. It frames the world view without covering the ordinary GUI.

Set the room’s Cinematic Override to Custom, enable Cinematic Presentation and adjust the controls. Use Inherit for the game-wide choice or Disabled to opt out. This is separate from Scene Shader Source.

Correct the lighting first. If a room is unreadably dark, adding exposure can reveal it, but it also lifts everything else. A useful cinematic grade refines a scene that already works.

> A Vignette preset and Cinematic Vignette can both be active. So can Film Grain in two places. Check for doubled treatments before blaming a slider.

17. Put a shader on one object

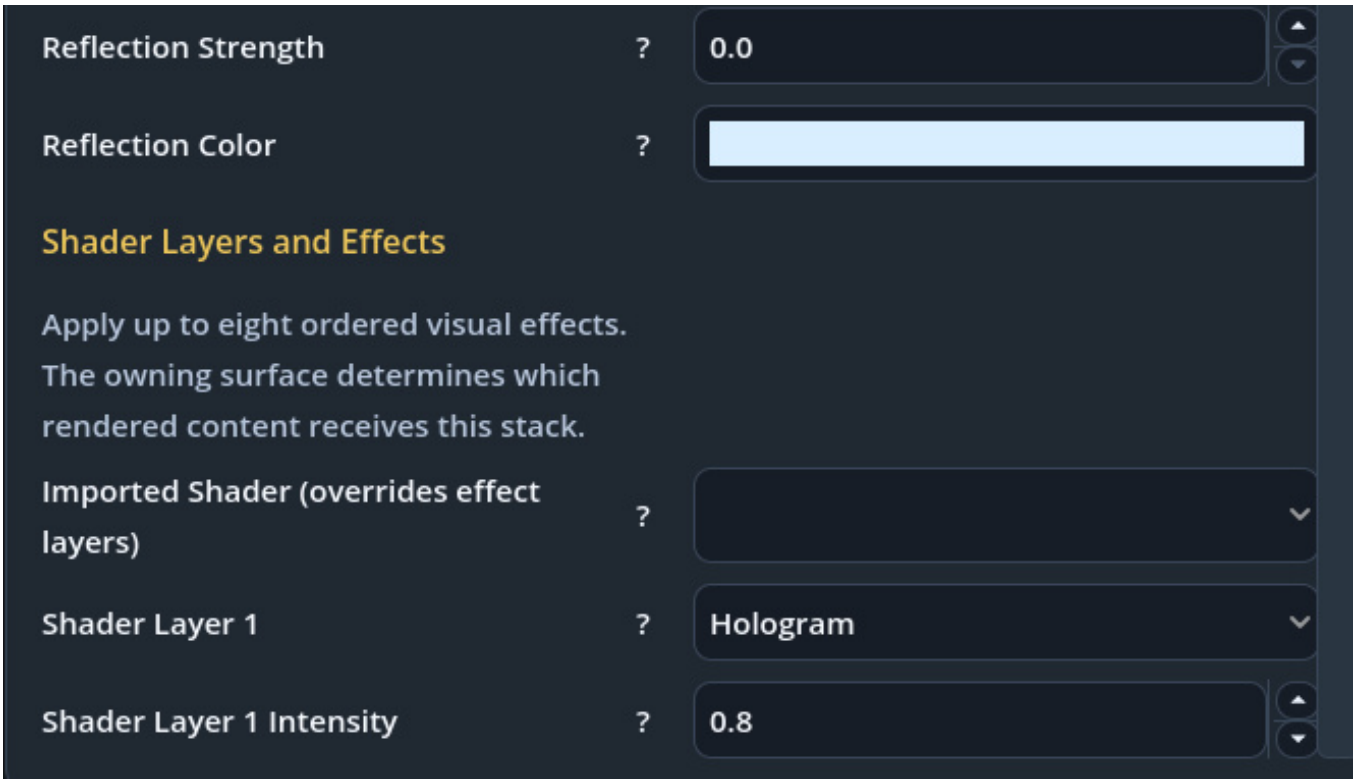

Figure 11. Shader Orb has its own surface stack. Its Hologram setting does not affect the other objects.

1. Select the object in Scene Objects. Open Visual Effects.
2. Scroll to Shader Layers and Effects. Leave Imported Shader empty for the built-in presets.
3. Choose Hologram in Shader Layer 1 and start at intensity 0.8.
4. Leave other layers at None, keep a neutral effect tint and start Animation Speed at 1.
5. Apply and test. Walk around the object or try another camera angle.
6. Set the layer to None to restore the ordinary appearance. The material’s existing color and texture remain the starting surface when no replacement effect is active.

Characters in a scene also have scene-specific Visual Effects. A reusable Character definition handles identity and animation; use the live scene instance when the change belongs only to this room.

A selected imported shader overrides the built-in stack. Certain specialized local presets, such as retro materials and Heat Haze, also replace the ordinary surface path rather than combining like generic color layers.

18. Make a glowing object light its surroundings

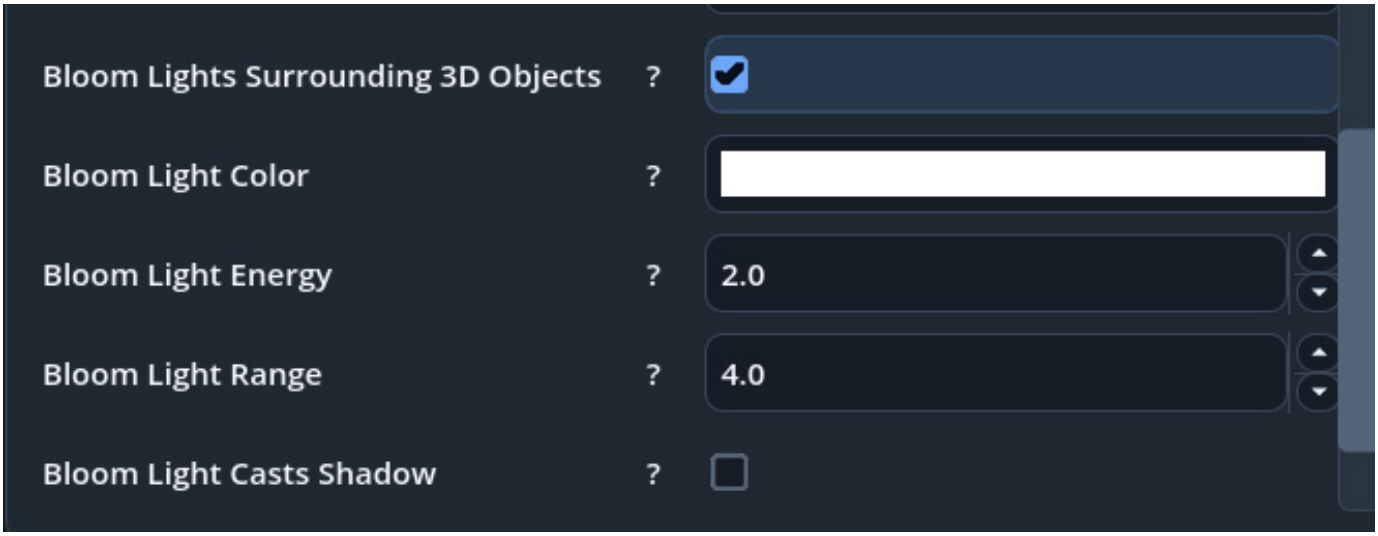

Figure 12. The object’s Bloom lighting controls add nearby illumination separately from the visible surface effect.

1. On a selected object, choose Bloom in a shader layer.
2. Find Bloom Lights Surrounding 3D Objects and enable it when the object should illuminate nearby scenery.
3. Choose a Bloom Light Color that matches the source.
4. Start with modest energy and range. A small crystal should not light the whole room.
5. Enable shadow casting only when the result needs it; then test its cost.
6. Check the room’s glow/bloom presentation and the selected renderer. A bright surface, a halo and a light pool are three things to judge separately.

Emission makes a surface look bright. A halo is the bloom/glow around bright pixels. A real nearby light changes the shading of the floor and neighboring objects. The explicit surrounding-light option creates that additional contribution for an authored object Bloom effect.

A lightning particle or fire preset does not automatically create this object light. For a flame that lights a wall, add a separate Light object, position or attach it appropriately, and give its motion or intensity a restrained variation if desired.

> The screenshots in this guide were captured with the Compatibility renderer. Judge advanced glow, depth of field and renderer-specific features on the backend and device you intend to ship.

19. Reflections: room capture versus a live mirror

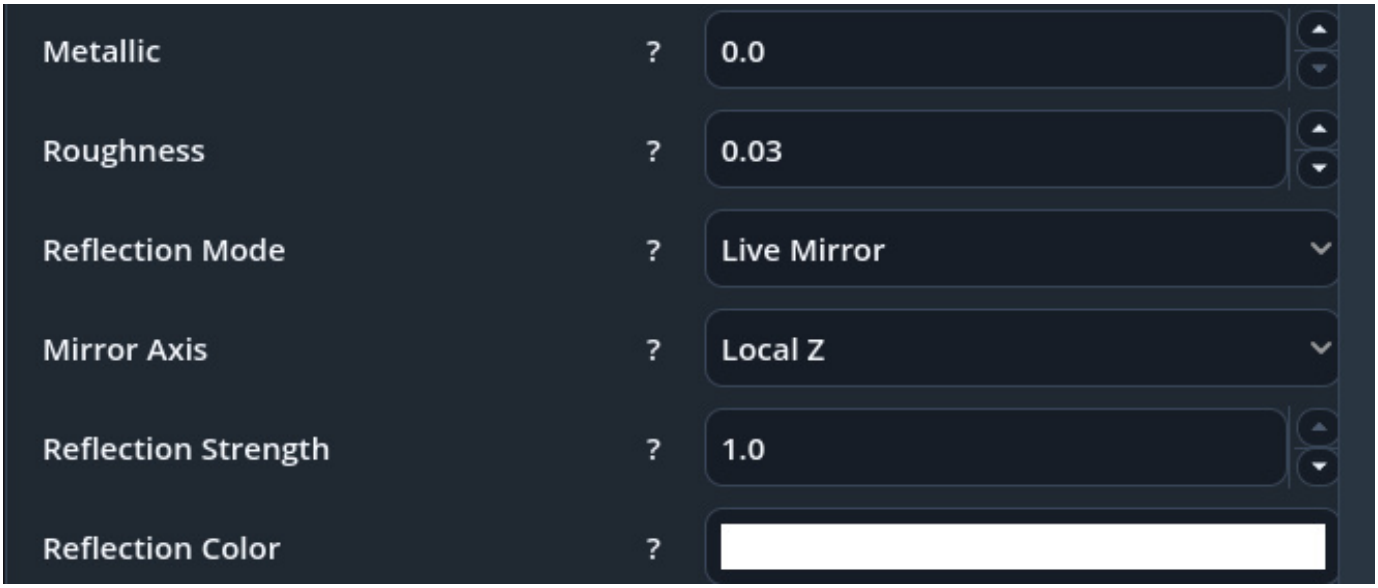

Figure 13. Live Mirror is an explicit Reflection Mode. Metallic and roughness alone are not a mirror setup.

1. Create a thin box or plane for a flat mirror. Give it a sensible physical size and face it toward the room.
2. Open that object’s Visual Effects → Material and Reflection.
3. Choose Live Mirror for Reflection Mode. Start Reflection Strength at 1, Reflection Color at opaque white and Roughness near 0.03.
4. Use Automatic (Thinnest Axis) for a thin box, or choose the correct local Mirror Axis. A wall mirror usually has a thin local Z dimension; a floor mirror usually has a thin local Y dimension.
5. Apply and use Test Scene. Put the character between the camera and the mirror plane, where a reflection should actually be visible.
6. If the result is wrong, first check the plane’s orientation and axis rather than raising metallic.

Room Capture gives an environmental reflection response. It is useful on polished objects, but it is a different job from a camera-correct character reflection on a flat mirror. Live Mirror uses an additional reflected view. Multiple visible mirrors therefore need a performance check.

> Use a flat surface for a clear mirror. Setting a curved character model or an entire elaborate prop to mirror mode is not a replacement for a carefully authored mirror plane.

20. Heat haze: screen, surface or particles?

Route	Use it for
Room Heat Haze preset	A shimmer over the finished room image, such as an extreme heat scene. Start Distortion Amount at 0.005-0.01.
Local object Heat Haze	A placed refractive plane above a stove or engine. It uses the scenery behind the surface.
Heat Haze particle emitter	A moving plume made of fading, rising refractive particles. Best for a small source that moves or needs an irregular outline.

1. For a local surface, add a Plane primitive and rotate it upright so the gameplay camera sees its face.
2. Place it above the hot source and size it to cover only the hot-air column.
3. Choose Heat Haze alone in its object shader stack.
4. Start intensity at 1, Distortion Amount at 0.02 and Animation Speed at 1.
5. Apply and test against a detailed background. Lower the amount if edges look like liquid glass.
6. To use a particle plume instead, follow the particle chapter and remove or disable the old helper surface after comparing.

The local surface is not painted fire. Its original surface color and image are not the visible content. Its UV-mapped surface defines the plume, and the shader refracts the opaque scene behind it. Avoid overlapping several heat surfaces and expecting cumulative physical refraction.

> Screen, surface and particle heat have different property names. The object uses shader.distortionAmount; the particle uses particle.heatDistortion. Setting one does not control the other.

21. Tint or blur a region of the scene

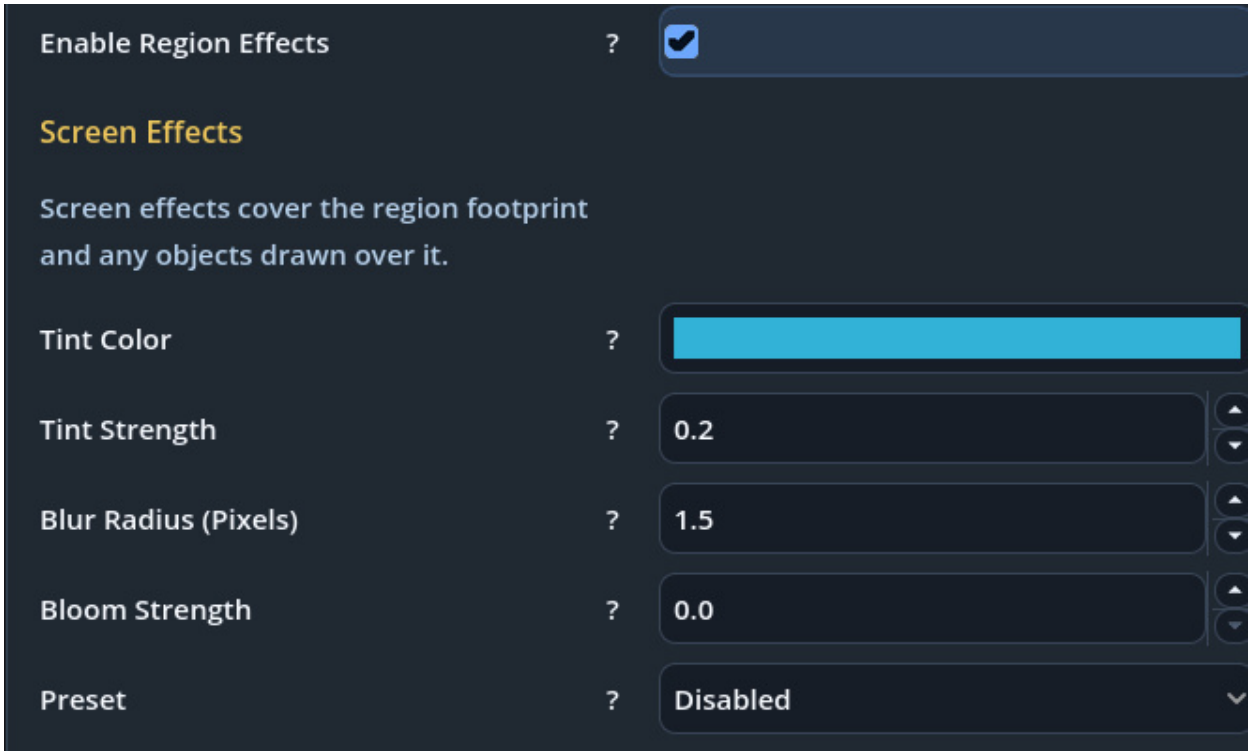

Figure 14. A Region can have screen effects over its footprint and a separate material, mirror or custom-shader surface.

1. Add or select a Region in the scene. Draw or size its footprint where the treatment belongs.
2. Open Visual Effects and enable the region’s effects.
3. Begin with Tint Strength around 0.2 and a distinctive color. Apply and test the footprint.
4. Add a small Blur Radius or Bloom Strength only if needed.
5. Choose a built-in Preset for a region screen treatment, or a Custom Screen Shader for a replacement screen effect.
6. For a physical-looking floor patch, set Surface Type to Material, Mirror or Custom Shader and configure that separate surface.

A region’s screen treatment covers the projected footprint and things drawn over it. It does not identify and recolor only the floor texture. This is useful for a magical pool or distorted patch, but a character in that footprint can receive the same screen effect.

Surface Height Offset lifts the added surface slightly above the floor. Use only enough to avoid flickering between coplanar surfaces. The Material surface has color, metallic, roughness and emission; Mirror has reflection strength; Custom Shader needs a spatial shader asset.

> Screen shaders are canvas effects. Surface shaders are spatial effects. Selecting a shader for the wrong kind of destination is not a way to convert it.

22. Import a shader without guessing its type

Use a built-in preset when it does the job. Import a shader when you have a specific effect and understand its inputs. A shader written for another engine or a shader website is not automatically ready to run.

1. Create or select an asset in Asset Designer and set its kind to Shader.
2. Browse to a .gdshader, .glsl, .frag or .fs file. Read the import result before applying.
3. For Godot source, the file must begin with shader_type spatial; or shader_type canvas_item;.
4. Choose a spatial asset for a 3D surface. Choose a canvas asset for a room screen shader, GUI or overlay.
5. Apply the asset and assign it through the destination’s Imported Shader selector.
6. Save and test in the intended renderer. Keep the source license and attribution with your project.

Importer boundary	What to do
Simple GLSL fragment with void main()	The limited converter can translate a predictable fragment subset.
Canvas GLSL fragment	The converter recognizes the pca3d:canvas marker. Otherwise GLSL imports as spatial.
Vertex program, includes or Shadertoy mainImage	Convert it manually to a compatible Godot shader first.
Custom uniform controls	Do not assume the built-in preset sliders populate arbitrary uniforms. Use suitable defaults or a deliberate custom setup.
A shader selected on a surface	It replaces that surface’s built-in effect layers while selected.

A shader can compile and still be wrong for its destination. A room screen shader needs to read the rendered screen; an object shader needs the surface’s texture or material inputs. A plain ColorRect does not supply a model’s texture automatically.

23. GUI shaders and world effects stay separate

A room screen stack normally affects the world image. The gameplay GUI remains clear above it. This is useful for heavy pixelation or fog: the player can still read dialogue and find the inventory controls.

1. Open GUI Designer and select the GUI or element you want to treat.
2. Use GUI Window for the whole authored window, or Element Shader for the selected control.
3. Choose one effect and test the whole control. A control shader can affect its text and artwork together.
4. Keep hover, clicked and background images configured in their normal Image controls. A shader does not replace those state assignments.
5. Use Preview for the control’s look; use Play for the complete room, GUI order and input behavior.

Goal	Suggested choice
Clear text over a retro room	Apply the retro effect to Scene Shaders; leave the GUI untreated.
A holographic device panel	Treat the panel or its decorative image, while keeping small text readable.
A magical icon only	Use the icon/element’s own shader rather than the whole HUD.
A temporary radial impact	Use Zoom Blur; leave Include GUI in Effect off initially.
A frame that shares the room style	Use Screen Overlay → Overlay Shader Source → Use Current Room Shaders.

Screen overlays sit below gameplay GUIs and ignore mouse input. A GUI is the appropriate tool when the artwork needs buttons, interactions, independently animated controls or several separately controlled layers.

24. Add your first particle emitter

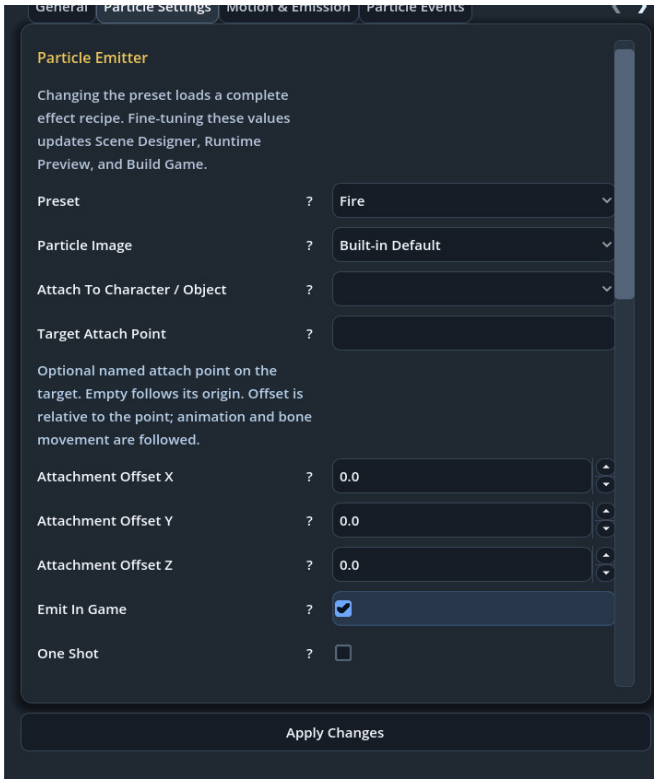

Figure 15. A Particle scene object has its own Particle Settings and Motion & Emission tabs.

1. In Scene Designer choose Add to Scene and add a Particle object.
2. Give it a recognizable name and stable ID, such as Brazier Flame and flame.
3. Select it in Scene Objects and open Particle Settings.
4. Choose the preset before fine-tuning. Start with Fire.
5. Place the emitter just above a brazier or other source using its Visual Adjustments position.
6. Apply Changes and use Test Scene. Check the effect through the actual gameplay camera.

An emitter is an ordinary scene record. It can be named, positioned, hidden, attached and controlled by events. Its particle settings are the recipe for its visible layers.

> Changing the preset loads that recipe’s defaults. Tune after selecting the preset. If you previously chose a custom Particle Image, check it too: changing the preset retains that image.

25. Choose a particle preset for the job

Preset	What it provides
Fire	Layered flame, sparks and smoke-like artwork for a burning source.
Snow	A broad field of drifting snow particles.
Rain	Fast falling streaks.
Hail	Small falling solid-looking particles. It does not add physical hail collision.
Wind	A directional moving particle effect.
Tornado	A rotating column-like visual effect; it does not pull characters into it.
Heat Haze	Rising refractive particles without painted fire or smoke.
Lightning	One procedural jagged bolt with timing, thickness and length controls.
Clouds	Soft drifting puffs; suitable for a cloud bank or low mist.
Storm Clouds	Darker and denser cloud defaults. Add rain/lightning separately.
Bug Cloud	Small wandering insect silhouettes.
Lightning Ball	Five crackling arcs around a glowing particle halo.
Explosion	A burst combining fireball, sparks and fading smoke.

These are the standard world-space Particle presets. They work in ordinary 3D rooms and the world-stage workflow. Older native Canvas/Particle2D emitters use a separate set of image and pixel-motion controls.

> The preset supplies visuals. Use actions, regions, triggers and scripts for damage, quest changes, collisions, camera shake and audio. A dramatic effect should not be expected to perform those jobs by itself.

26. Amount, lifetime and one-shot timing

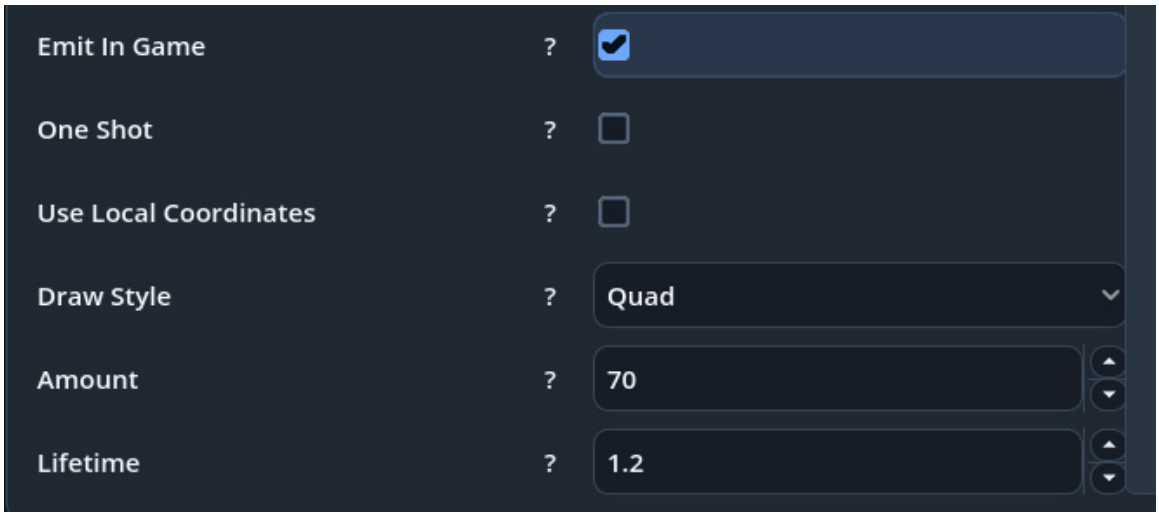

Figure 16. The emitter’s emission, count and lifetime controls are separate from its visible artwork.

Control	Meaning
Amount	The base density/count used by the recipe. Multi-layer presets can create more than one set of particles.
Lifetime	How long a particle or recipe layer remains. Different layers can multiply this value.
Preprocess	Simulates some time before the effect appears, useful for an already-filled looping cloud bank.
One Shot	Produces one emission cycle instead of a continuous loop.
Explosiveness	Concentrates emission toward a burst rather than spreading it evenly through the cycle.
Lifetime / Emission Randomness	Varies timing so the effect is less uniform.
Emit In Game	Enables the effect’s runtime emission. Switch off and back on to restart a supported burst.

For a new blast use One Shot on, Explosiveness 1 and Preprocess 0. For continuous fog use a looping cloud recipe with low velocity and a restrained preprocess value.

> A preset preview is a working editor aid, not a frame-accurate recording of the first gameplay second. Some one-shot previews repeat, and preview particle counts can be reduced. Always restart Test Scene to judge the actual one-shot timing.

27. Shape particle motion



Figure 17. Motion & Emission separates movement controls from the particle’s artwork and color.

Control	How it changes the source
Direction	The initial axis of travel. [0,1,0] points up; [0,-1,0] points down; [1,0,0] points along X.
Velocity Min / Max	The starting speed range. Keep Min no greater than Max.
Spread	How widely the particles depart from the initial direction.
Gravity	Acceleration during life. Negative Y falls; a positive Y value can lift a plume.
Turbulence	An additional irregular influence. Use small strength values for slow drifting particles.
Turbulence Scale	The scale of the irregular motion pattern. Its appearance can differ by renderer.

Emitter rotation also changes the orientation of local directions. If a bolt or plume points sideways after attaching it to a hand, inspect the hand point’s rotation as well as Direction.

> A preset is a multi-layer recipe, not one universal particle material. Individual layers can use their own multipliers or motion choices. Lightning geometry ignores the ordinary velocity, spread and gravity controls; use its bolt controls instead.

28. Size the emission area and artwork separately

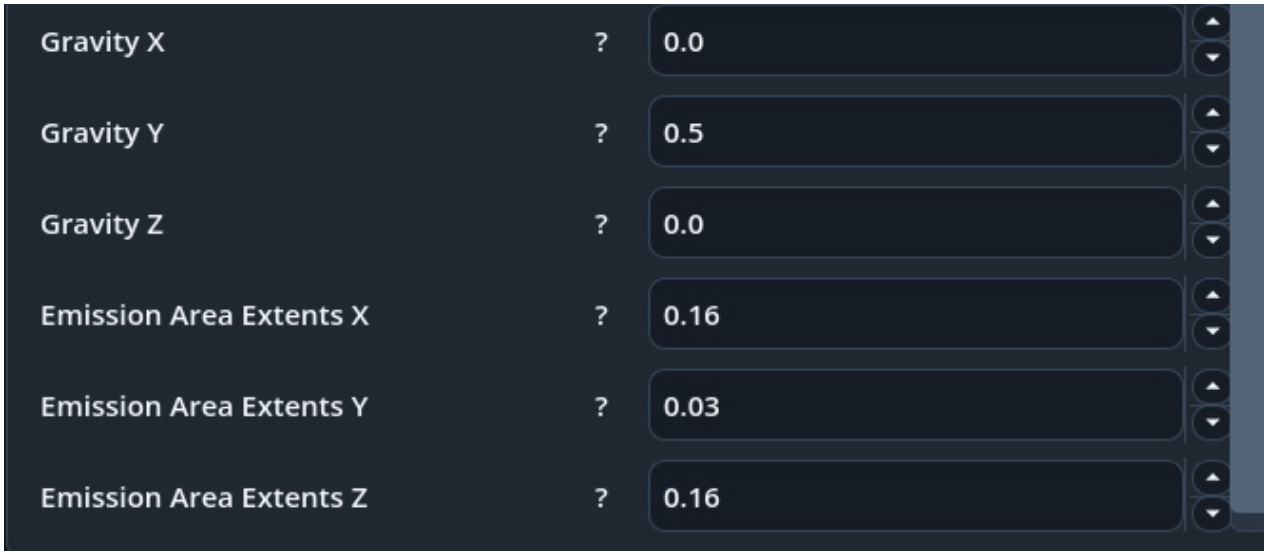

Figure 18. Direction, gravity and the three Emission Area Extents are numeric vectors.

Particle Size changes the visible pieces. Emission Area Extents changes the volume where new pieces begin. Enlarging every flake is not the same as covering more of the room.

Result	What to change first
A tight flame above a cup	Keep X/Z extents small and use modest Particle Size.
Snow across an outdoor scene	Raise the emitter above the room and widen X/Z extents.
A thin mist bank near the floor	Use wide horizontal extents, a low vertical extent and lower opacity.
A narrow exhaust plume	Use a small source area, an upward direction and a narrow spread.
A bigger individual cloud puff	Raise Particle Size rather than widening only the source area.

For a box-shaped emission area, think of extents as distances from its center. An X extent of 3 describes a span approximately 6 units wide before later motion. Velocity and lifetime can carry particles beyond that initial area.

> Use actual room scale. An emitter designed for a 100-unit landscape can swallow a 2-unit-tall character. Start small and widen it while watching the gameplay camera.

29. Give particles a custom image

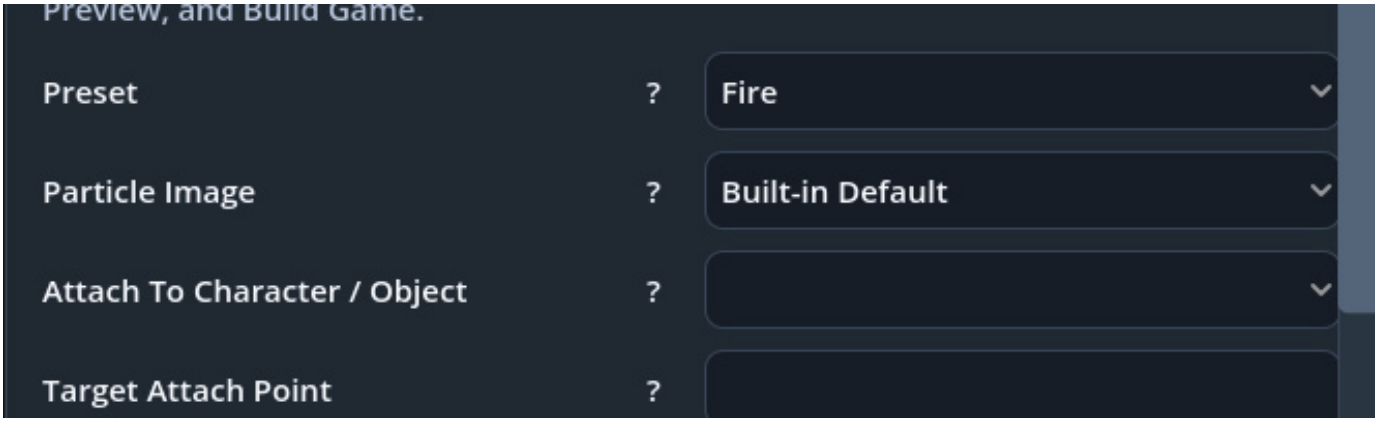

Figure 19. Particle Image uses an ordinary imported image asset. Built-in Default restores the generated artwork.

1. Import a transparent PNG, or another supported image, through Asset Designer as an Image asset.
2. Select the Particle object → Particle Settings → Particle Image.
3. Choose the image by its name and ID.
4. Start Primary Color and Emission Color at white if you want to preserve the artwork’s colors.
5. Adjust Particle Size and Opacity, then apply and test.
6. Choose Built-in Default to compare with the preset’s generated image.

The image keeps its aspect ratio and alpha. Image-based particle layers use camera-facing quads. Recipe color gradients and lifetime fades still apply. A multi-layer effect such as Explosion can use the chosen image on several layers, so a very specific shape may look strange as smoke.

Heat Haze retains its refractive material. Lightning arcs retain their geometry. Lightning Ball uses the custom image for its halo, not for the five arcs. This selector uses a static image; it is not a sprite-sheet animation timeline.

> Changing presets retains the selected Particle Image. Clear that image if the next preset mysteriously keeps looking like the previous one. Missing image references should also be checked in Project Validation.

30. Build a brazier from two effects



Figure 20. The brazier uses visible Fire and a separate Heat Haze plume. Each can be tuned or switched independently.

1. Create or place the brazier prop. Keep its physical geometry separate from its effects.
2. Add Fire above the bowl. Start with Amount 70, Particle Size 0.22 and Lifetime 1.2.
3. Use a small Emission Area, such as [0.16,0.03,0.16], and modest upward speed.
4. Add a second Particle object named Brazier Heat and choose Heat Haze.
5. Place it a little above the flame. Try Amount 16, Size 0.5, Opacity 0.35, Heat Distortion 0.015 and Heat Motion Speed 0.8.
6. Apply both records. Test with a detailed wall or character behind the plume so the distortion can be seen.
7. Add a separate warm Light object if the flame should illuminate nearby surfaces.

Keep the heat plume subtle. Too much distortion can make the brazier look underwater. If nothing appears, compare with the emitter disabled, then look at a sharp background edge through the hot area.

> The flame’s built-in bright artwork is not an automatic real light. The heat emitter adds no orange tint, smoke, light or sound. Those are separate choices.

31. Make rain, snow and hail cover the room



Figure 21. World-space snow starts above the room. Its source area and fall direction determine the coverage.

1. Add one Particle object and select Snow, Rain or Hail before adjusting its settings.
2. Move the source above the visible room. Check the transform’s Y component.
3. Widen its X/Z Emission Area to cover the camera’s visible ground.
4. Keep Direction downward. Snow can use slow velocity and gentle negative-Y gravity; rain and hail use faster recipes.
5. Restart Test Scene. Watch both the upper view and the floor: a source too low can appear only around the character’s feet.
6. If the scene contains a roof, do not expect particles to know it is there. Place or limit emitters deliberately.

A weather emitter has no automatic indoor/outdoor masking or physical precipitation collision in this workflow. Use separate emitters for different spaces, turn them on and off when appropriate, or use a localized screen/region effect for a particular view.

> Hail is visual. Add explicit interaction logic if hail is supposed to hurt a character. Particle snow also does not trigger the separate Snow Surface Accumulation coating.

32. Clouds, floor mist and a small swarm



Figure 22. Clouds has a longer lifetime and broad area. Start with the recipe, then reduce it for a small room.

Example	Suggested changes
Distant cloud bank	Keep velocity low, use large puffs and a broad source. Raise the emitter into the sky.
Low floor mist	Flatten the emission area, lower its position and opacity, and keep drift slow.
Storm cloud	Choose Storm Clouds for its darker recipe. Add separate Rain and Lightning objects as needed.
Distant insect swarm	Try Amount 25, Size 0.04 and an area around [0.5,0.35,0.5].
Firefly-like motes	Choose a soft bright image and appropriate colors. Add real lighting only where the scene requires it.

Cloud colors are drawn artwork rather than a volumetric sky simulation. They do not automatically respond to every room light or cast cloud shadows. Large translucent puffs can be costly even when the particle count is small.

Bug Cloud supplies visible movement; it does not create a collection of independently colliding insect characters. For an insect that can be clicked or used in a puzzle, create an ordinary interactive entity in addition to the background swarm.

> Preprocess helps fill a looping bank before it appears, but high values increase startup work. Use only enough to establish the intended density.

33. Author a lightning bolt

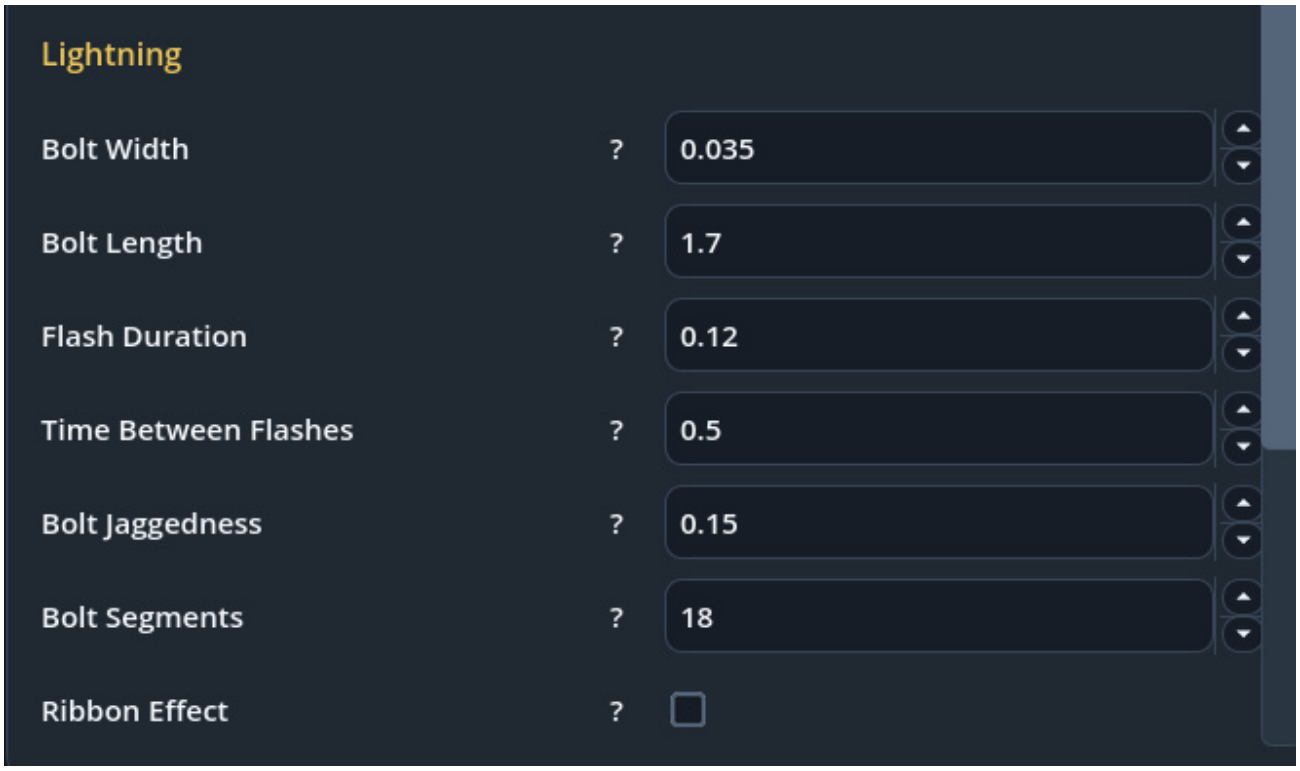

Figure 23. Lightning has dedicated bolt controls. Amount and ordinary particle speed do not shape the bolt.

Control	Meaning
Bolt Width / Bolt Length	Outer thickness and source-to-tip distance in scene units.
Flash Duration	Seconds for one visible flash.
Time Between Flashes	Dark interval after a flash. Zero makes a continuous beam.
Bolt Jaggedness / Segments	Sideways irregularity and the number of line segments.
Primary / Emission Color	The outer bolt and bright core.
Ribbon Effect	A camera-facing flat strip instead of volumetric bolt geometry.
One Shot / Emit In Game	One flash, or repeating flashes; disable and re-enable to start again.

For a small spark try width 0.025, length 0.5, duration 0.05, interval 0.2 and 12 segments. For an important cue at low frame rates, make the flash longer so it is not missed.

> Ribbon is not a trail. Moving the emitter moves the whole bolt. The bolt has no automatic sound, damage, collision, branching or scene light. Direction and the emitter/attach-point rotation define its orientation.

34. Put a lightning ball above a hand



Figure 24. A small Lightning Ball follows Ivo’s right-hand attach point during play.

Control	Ball-specific meaning
Particle Size	The ball’s diameter, not straight-bolt length. Try 0.38 for a small hand effect.
Amount	Halo particle density. It does not change the five-arc count.
Bolt Width / Jaggedness / Segments	The appearance of the arcs around the ball.
Flash Duration / Time Between Flashes	The arc timing. The halo still follows its own particle life.
Ribbon	Flat arcs facing the active camera; still not a movement trail.
Particle Image	Replaces halo artwork while the arcs remain procedural.

Create the ball as a Particle object first. Give it a small size, then attach it to an existing hand point using the later attachment instructions. For an ability that is initially inactive, clear Emit In Game and turn it on from the ability’s action list.

> The ball’s arcs always follow the emitter transform. Use Local Coordinates affects ordinary particle behavior around it; it does not leave a detached copy of the arcs behind a moving hand.

35. Trigger an explosion more than once

An Explosion is a visible burst. It can be combined with sound, a light flash, camera shake and gameplay consequences, but those are separate actions. The recipe’s smoke can last longer than the initial fireball.

1. Create a Particle object named Single Explosion, ID blast, and choose Explosion.
2. Leave One Shot on, Explosiveness at 1 and Preprocess at 0.
3. For a small impact, start Amount 50, Size 0.12, Velocity Max 2 and Lifetime 0.5.
4. Clear Emit In Game if the blast should wait for a trigger. Apply and save.
5. In that trigger’s action list, turn particle.enabled off and then on for blast.
6. Add Play Sound or a short Zoom Blur as separate steps. Restart Test Scene and trigger the event twice.

```
[
  {"type":"ChangeObjectProperty","entityId":"blast",
   "property":"particle.enabled","value":false},
  {"type":"ChangeObjectProperty","entityId":"blast",
   "property":"particle.enabled","value":true}
]
```

The smoke layer lasts roughly 1.6 times the selected Lifetime in the bundled recipe. Wait for the actual tail when deciding when to remove a prop, cut away or restore the scene.

Re-triggering restarts the effect; it does not create a separate independent blast instance each time. Use separately authored emitters when several overlapping blasts must continue independently.

36. Stop emission, hide an effect or detach it

Operation	What it means
particle.enabled = false	Stops runtime emission. Ordinary particles already alive can continue fading until their life ends. Specialized effects can stop/rebuild differently.
Set Object Visible = false	Hides the entity’s visual result. Use when the whole effect must disappear immediately.
Detach Particle Effect	Releases the follower link and keeps the emitter at its current world pose. It does not switch emission off.
Detach then disable emission	Leaves a source behind and stops new pieces, allowing a supported ordinary tail to finish.
Switch preset in the editor	Loads a new recipe. Do this while authoring rather than expecting arbitrary runtime conversion.

Choose the intended ending. A hand aura should usually disappear with the hand effect. Exhaust might stop producing new particles while its old wisps drift away. An explosion should normally finish its smoke rather than vanish at the end of the fireball.

1. Write down which entity owns the visual and which event turns it on.
2. Decide whether the end should stop emission, hide it, detach it, or combine those actions.
3. Test during movement as well as while idle.
4. For a cutscene, run the same cleanup after normal completion and after skipping.

> Removing the target or changing rooms is not a substitute for a clear effect lifecycle. A script that controls an effect should also define how the player gets back to the ordinary state.

37. Which particle changes update during play?

The shared property API can store many values, but an immediate visible rebuild is not guaranteed for every preset and field. Author the emitter and its preset first. Use this distinction when an action changes a value but the current particles look unchanged.

Authored preset / property	Current runtime behavior
All ordinary GPUParticles: particle.enabled	Updates emission; enabling restarts the particle system.
Image-based particle.imageAssetId	Updates supported sprite-layer artwork. Heat Haze keeps its specialized material.
Heat Haze: heatDistortion, heatSpeed, opacity	Refreshes the existing refractive particle material.
Lightning bolt fields	Refreshes the existing procedural bolt.
Clouds, Storm Clouds, Bug Cloud, Lightning Ball, Explosion	Live particle-property changes rebuild that effect’s layers while preserving its entity transform and attachment. Particle ages/timing restart.
Older ordinary recipes: arbitrary amount, speed, localCoords etc.	Do not assume an immediate rebuild. Apply authored settings and restart/re-enter the room to verify the new setup.

This makes property changes suitable for authored events, such as changing a ball’s size or starting a blast. It is not an invitation to rebuild a cloud bank on every frame. Rebuilding restarts its visible history and can cause a hitch.

> Use a variable or event to choose between already-authored effects when a dramatic change is needed. Repeatedly changing the preset string of an unrelated live emitter is not the supported setup workflow.

38. Keep particles affordable and readable

Particle cost is not only Amount. A few enormous translucent clouds can cover most of the screen and be more expensive than many tiny sparks. Overlapping transparent layers repeatedly draw the same pixels.

Problem	First useful adjustment
A room opens slowly	Reduce preprocess and very large effects; compare a clean room.
A cloud bank slows the view	Reduce puff size, overlap and opacity before simply doubling its count.
A lightning ball is too busy	Reduce segment count, width, halo density or flash frequency.
Fire hides an interactive prop	Reduce size/opacity and separate the bright source from the clickable silhouette.
The effect changes between renderers	Compare turbulence, refraction, glow and transparency on the target backend.
A distant one-shot is never noticed	Use a longer visible duration, stronger composition or a sound cue, not just more particles.

1. Check the default recipe before raising counts.
2. View the effect at the nearest and farthest gameplay distances.
3. Test a normal gameplay route with several effects visible together.
4. Build and run on the slowest device you intend to support.

> The screenshots prove the illustrated setup in the captured editor/runtime. They are not a performance certification for Windows, macOS, Android or a particular GPU. Test the finished game on those devices.

39. Attach points are named sockets

An attach point gives a useful name to a position and orientation on an entity. A hand can have right_hand_grip, an engine can have exhaust, and a torch can have flame_top. The names describe where an effect belongs rather than where the character happens to be standing.

Part of the link	Example
Source / child	The Particle object hand_orb. This is the thing that moves to follow.
Target / parent	The live player instance ada, also available as player when active.
Target Attach Point	right_hand_grip on the player.
Attachment Offset	A small adjustment measured relative to that point’s orientation.
Own Attach Point	A point on a carried prop when one prop point must meet the hand point. Particle attachment normally uses the emitter origin.

The link is updated as the target moves. A point bound to a skeleton bone also follows that bone’s animation. A point that is merely near the hand in object space follows the character root, but it will not stay in the hand when the arm moves.

A surface shader travels with its object: attach that object if a glowing prop should follow a hand. A screen shader belongs to the view and cannot be socketed onto a bone. For a local magical effect, attach an ordinary shader-covered object or an authored particle emitter.

> Create the target entity and point before linking the emitter. A name that looks right in a text box is not proof that a point with that exact name exists on the live target.

40. Place a point in Asset Designer

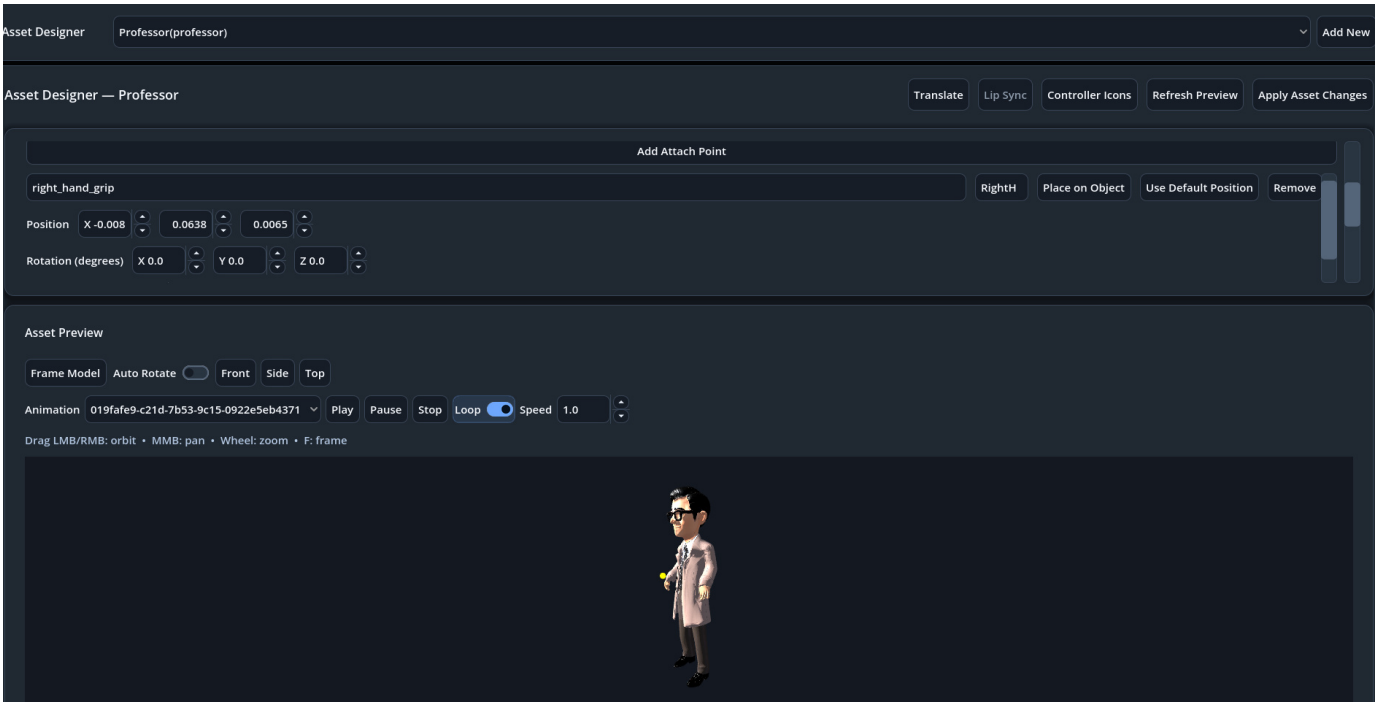

Figure 25. Asset Designer combines the model preview with the Attach Points editor and animation scope.

1. Open the target model in Asset Designer.
2. Frame the model so the relevant part is visible. Stop automatic rotation if it makes placement difficult.
3. In Attach Points, choose All Animations for the base point.
4. Press Add Attach Point, then click the actual model surface in the preview. The preview pauses for placement.
5. Rename the point to something exact and useful, such as right_hand_grip. Inspect the Position, Rotation and optional Bone Name fields.
6. Use Place on Object to reposition the selected point. Clicking empty space does not establish a useful new location.
7. Apply Asset Changes and save the project.

Yellow markers identify the points. For a non-animated machine, an object/model-space point is often enough. For a skinned hand, inspect the chosen bone and test a moving animation before accepting the point.

> Do not guess a bone name from another model. One imported skeleton may use RightHand while another uses mixamorig_RightHand. Use the actual names and preview of your asset.

41. Bind the point to the moving bone

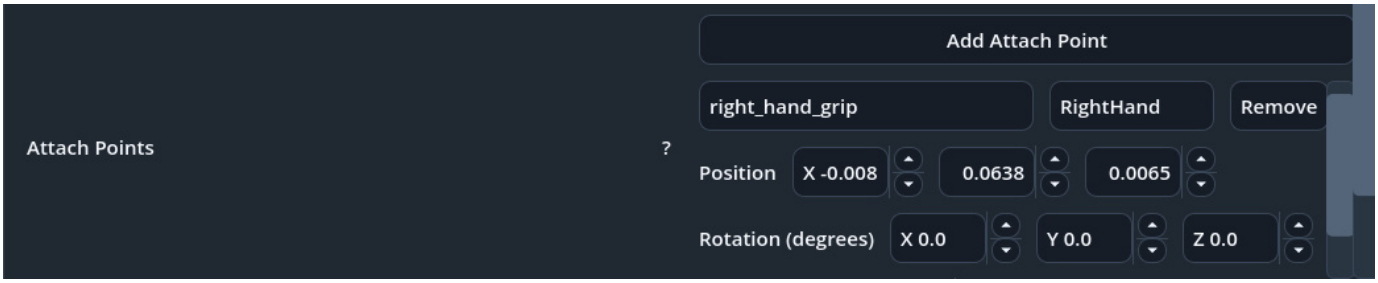

Figure 26. The scene instance exposes its named hand point, bone binding and numeric offsets.

1. Select a suitable idle or hand pose in the model preview.
2. Place the point on the hand and inspect Bone Name. On a skinned model the picker can choose the strongest contributing bone at the clicked surface.
3. Play a walk or gesture animation. Watch whether the marker stays with the hand.
4. If it follows the torso instead, correct the bone binding or place it again on the intended skinned surface.
5. Use a small position adjustment for the grip or emitter center. Use rotation to define the direction a carried prop or bolt should face.
6. Apply and repeat the test in the real scene.

For a scene instance in this build, Attach Points is near the bottom of the 2D / 2.5D tab, even when the character itself is used in a 3D room. Asset Designer remains the convenient place to place a model socket visually.

The hand point is not an animation. It follows animation that already exists in the model. It does not create a grip pose, reach toward an object or solve the arm with inverse kinematics.

If the model is re-exported with different bone names or scale, recheck its points. A good-looking static point can become inaccurate after a substantial model change.

> Attach-point contact radius is measured in world units in 3D, or pixels in older native 2D. It does not automatically scale with the model. Its purpose is contact queries, not particle plume size.

42. Use per-animation point adjustments sparingly

All Animations defines the base point. A particular imported clip can override its position, rotation or bone when that clip needs a different offset. This is useful for a model whose source clips do not share exactly the same grip alignment.

1. Get the All Animations point working first.
2. Choose one imported clip in the attach-point animation selector.
3. Play or pause at a useful pose and adjust that clip’s point only if it needs a correction.
4. Switch back to All Animations and check that the base definition still makes sense.
5. Try another clip without an override. It should use the base point.
6. Use Use Default Position to remove the selected clip’s special override, then apply the asset.

Setting	What it is not
A per-clip offset	It is not a set of point keyframes through the clip.
A bone binding	It is not a new animation channel or an IK solver.
A point rotation	It does not repair an incorrectly animated wrist.
All Animations	It is the fallback definition for clips without a specific override.

A clip override takes precedence over the base point. If changing the base position seems to do nothing in one animation, inspect that animation’s override before making a large correction.

43. Know which record owns the point

The same named point can be authored on an asset, a Character definition, a scene instance or the running instance. Use that deliberately: shared points should be shared, while one-off adjustments should stay local.

Layer, in priority order	When to use it
Asset point	A reusable socket that belongs to every use of that model or image.
Character point	A reusable character-specific correction over its model’s points.
Scene-instance point	An adjustment for this character/object in this room.
Runtime edit	A change made by the running game, affecting that live instance and its saved room state.

Later layers override matching names from earlier layers. Renaming a point creates a different point; it does not rename every attachment that referred to the old name. Use exact spelling and case consistently.

1. Find the point on the record that should own it.
2. If it belongs to every instance, prefer the shared asset or Character definition.
3. If only one scene needs an adjustment, edit the instance.
4. After applying, check the live target rather than assuming that the visible shared point wins over every local override.

Runtime point additions, edits and removals do not rewrite the original project file. An instance removal can mask an inherited point. Saved runtime state and the source project are separate things.

44. Attach the authored particle in the Inspector

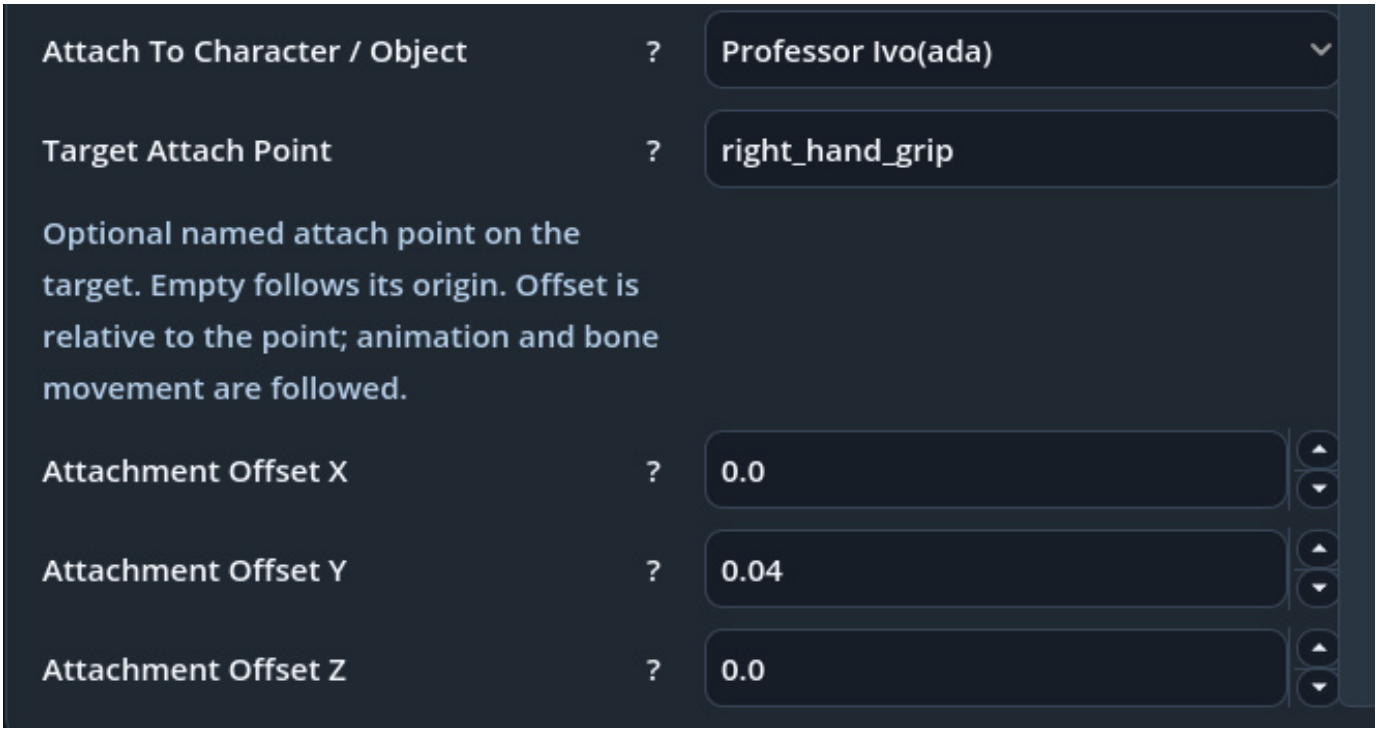

Figure 27. Hand Lightning Ball targets the live Professor Ivo instance and its right_hand_grip point.

1. Create the Particle object and choose Lightning Ball first.
2. Set Particle Size to about 0.38 for the workshop hand effect.
3. In Particle Settings, choose Professor Ivo(ada) under Attach To Character / Object.
4. Enter right_hand_grip as Target Attach Point.
5. Start Attachment Offset at [0,0.04,0]. Use zero first if the direction of that small offset is unclear.
6. Apply Changes and save. Start Test Scene and enable emission if it is initially off.
7. Make Ivo walk and play a gesture. The ball’s center should follow the hand rather than its original room position.

A blank Target Attach Point means the target’s origin. For a character, that is often near the feet. If the effect appears there, check whether the point field is blank or misspelled.

> The source is the particle. The target is the character. Reversing those roles would ask a character to follow an effect, which is a very different operation.

45. Fix position and direction after attaching

An attached effect is positioned from the target point every frame. Dragging its old world position is therefore not the reliable way to correct the final placement. Correct the target point or its Attachment Offset.

Symptom	First correction
The effect is consistently above the hand	Reduce the offset on the appropriate local axis. Start from [0,0,0] if necessary.
The effect rotates oddly as the arm moves	Inspect the point rotation and bone choice.
A bolt comes out sideways	Check its Direction and the target point’s rotation together.
The effect follows the body, not the wrist	Use a bone-bound point rather than a root-space point merely placed near the hand.
One animation is wrong	Inspect that clip’s point override.
The target selector names the wrong actor	Choose the live scene entity; verify the stored ID.

1. Freeze the actor in an easily understood pose.
2. Start with zero attachment offset.
3. Change one axis a small amount and test. This reveals which way that point’s local axis faces.
4. Check another animation before finishing. A point that works only while idle is not yet a reliable hand socket.

> Attachment Offset is relative to the point’s orientation. “Positive Y” there is not necessarily the same visible direction as the room’s up axis after the hand rotates.

46. Local coordinates decide what old particles do



Figure 28. The attached emitter follows the moving actor. Ordinary emitted particles can either travel with it or remain in room space.

Use Local Coordinates	What you should expect
On	Ordinary live particles stay in the emitter’s local space. Moving the emitter carries the existing group along. Useful for a compact aura.
Off	Particles already emitted occupy the room’s space while the source moves on. Useful for exhaust or a drifting trail.
Lightning / Lightning Ball arcs	The procedural geometry follows the emitter transform. Ribbon does not turn it into a trail.

The attachment and the coordinate setting answer different questions. The attachment decides where new particles start. Local Coordinates decides how ordinary particles that already exist relate to a moving emitter.

1. Use a clearly visible cloud or mote emitter with a lifetime long enough to observe.
2. Attach it to a moving target and test with Local Coordinates on.
3. Apply the opposite setting and restart the test. Do not assume every older preset updates that flag immediately on an existing particle system.
4. Compare the particles after the actor has moved several body widths.

> A trail is made from particles left behind the source. A camera-facing Ribbon lightning bolt is one moving strip. These are not the same effect.

47. Attach and detach through an action list

1. Create the effect and target point in the editor first. Save both.
2. Open the event that should activate it, such as a GUI button, verb interaction or trigger.
3. Choose Attach Particle Effect from the shared Action List catalog.
4. Set Particle Effect to hand_orb, Target Object to ada (or player), Target Attach Point to right_hand_grip and Attachment Offset to [0,0.04,0].
5. Add Change Object Property with entity hand_orb, property particle.enabled and value true.
6. On the ending event, use particle.enabled false or hide the source as required, then Detach Particle Effect when the link should end.

```
{
  "type":"AttachParticleEffect",
  "entityId":"hand_orb",
  "targetId":"player",
  "targetPoint":"right_hand_grip",
  "offset":[0,0.04,0]
}
```

These are the same shared actions in visual nodes, cutscene events and text scripts. The attachment action moves an existing emitter; it does not create a new Particle object.

Detach keeps the emitter’s current world pose and does not disable it. If an aura should disappear when released, define that visibility/emission step explicitly.

48. Carry a prop and attach an effect to it

For a staff with a glowing tip, make two links: the staff follows the hand, and the particle follows the staff’s tip. This keeps the effect tied to the prop rather than making it imitate a guessed hand offset.

1. On the character, create a bone-bound right_hand_grip point.
2. On the staff, create a grip point where the hand should hold it and a tip point at the glowing end.
3. Place the staff as an ordinary live scene object.
4. Use Attach Object to align the staff’s grip point to the character’s right_hand_grip point.
5. Use Attach Particle Effect to attach the authored particle to the staff’s tip point.
6. Test idle, walking and gesture animations. Fix the staff’s grip alignment before adjusting its tip effect.

```
await AttachObject({"entityId":"staff","point":"grip",
    "targetId":"player","targetPoint":"right_hand_grip"})
await AttachParticleEffect({"entityId":"staff_glow",
    "targetId":"staff","targetPoint":"tip",
    "offset":[0,0,0]})
```

The attachment system rejects self-links and cycles. Do not attach the staff to its own particle or attach the character back to the staff. All participants must be compatible live entities in the loaded room.

> A prop’s Own Attach Point and the target’s Target Attach Point are different fields. The first belongs to the child being moved; the second belongs to the destination.

49. Point contacts and attachment are separate

A contact radius around a named point can test whether two points touch. This is useful for a hand reaching a switch or two authored hand points meeting during a handshake. Attaching an emitter does not require a nonzero contact radius.

Operation	Purpose
Get Attach Point	Inspect the current world point and its resolved definition.
Attach Points	Align two points and keep following. Its movement-suspension option is useful for actors, not needed to animate a particle source.
Align Attach Points	Align once, without retaining a follower link.
Attach Points Touching	Test the current pose against point radii and optional tolerance.
Watch Attach Point Contacts	Watch a named entity pair for contact callbacks.
Remove Attach Point	Remove/mask an instance point and release links that depended on it.

For attach/get calls, a blank point can mean the entity origin. For contact queries, a blank point filter means all enabled named points on that side. It does not invent an origin contact sphere.

Contact queries do not apply particle damage automatically. If a lightning effect should activate a machine when the hand reaches it, use the contact result or callback to run the machine’s action list.

> Do not use an actor-to-actor persistent attachment as a shortcut for every interaction. It can move the source actor. Choose contact testing, one-time alignment or a persistent link according to the behavior you actually need.

50. What parallax actually needs

Parallax is relative movement between layers. A distant skyline seems to move slowly as you walk; a nearby railing moves faster. In an adventure scene you can suggest that depth with separately painted images and different scrolling factors.

Requirement	Why it matters
A moving camera	No camera movement means no camera-driven parallax. Moving the mouse alone does not demonstrate it.
Separate images or cutouts	One flattened background cannot move its painted trees independently from its sky.
Enough spare artwork	Scrolling can reveal an edge if a layer is only as wide as the initial view.
A chosen stacking order	Opaque artwork in front can hide the layers behind it.
Different factors	If all layers use the same factor, they move together and do not suggest separate depths.

Current 2D and 2.5D Stage rooms use world-space planes across X/Z with a locked orthographic camera. A full 3D room already has perspective parallax from real geometry; screen overlays can add a separate decorative camera-driven layer.

> Parallax does not extend navigation, resize an event volume or change the player’s collider. The camera bounds, image dimensions and walkable area remain separate settings.

51. Prepare a scrolling 2D Stage

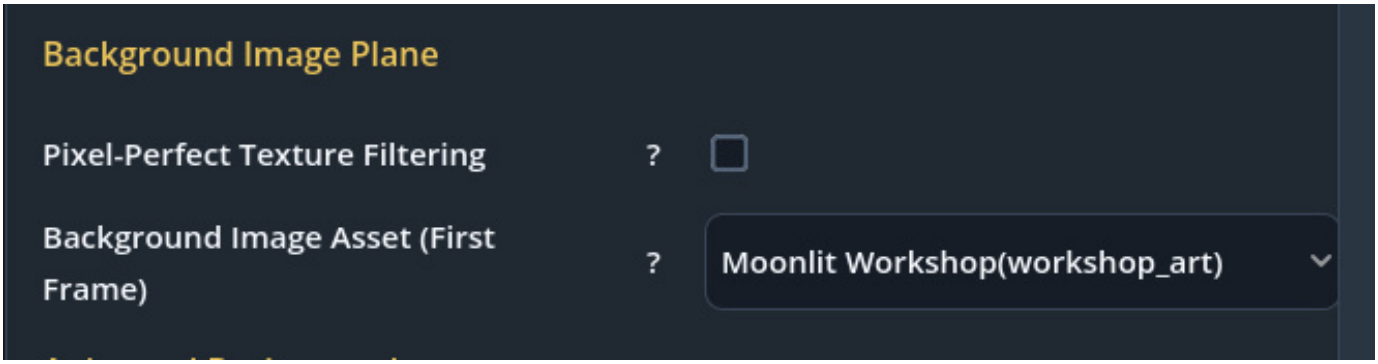

Figure 29. The stage background has a world size. This is separate from the camera’s visible height.

1. Create or select a current 2D or 2.5D Stage room.
2. In the room Inspector, open 2D / 2.5D Room and assign the background image.
3. Set the background’s world width and height. For the illustrated example, use a 32-by-18-unit stage.
4. Keep the source image’s aspect ratio unless you deliberately want stretching.
5. Set the ground and navigation for the area the character should walk across.
6. Apply, save and confirm that the plain background appears in Test Scene before adding layers.

A 16:9 camera showing 11.25 world units in height sees roughly 20 units in width. A 32-unit-wide background therefore leaves room to pan horizontally. If the view covers the whole background, there is little or no useful scroll range.

> Increasing Source Pixels per World Unit is not the same as widening the stage. Size the image plane and navigation explicitly. A visually large background does not automatically make a larger walkable floor.

52. Set the camera’s scroll range

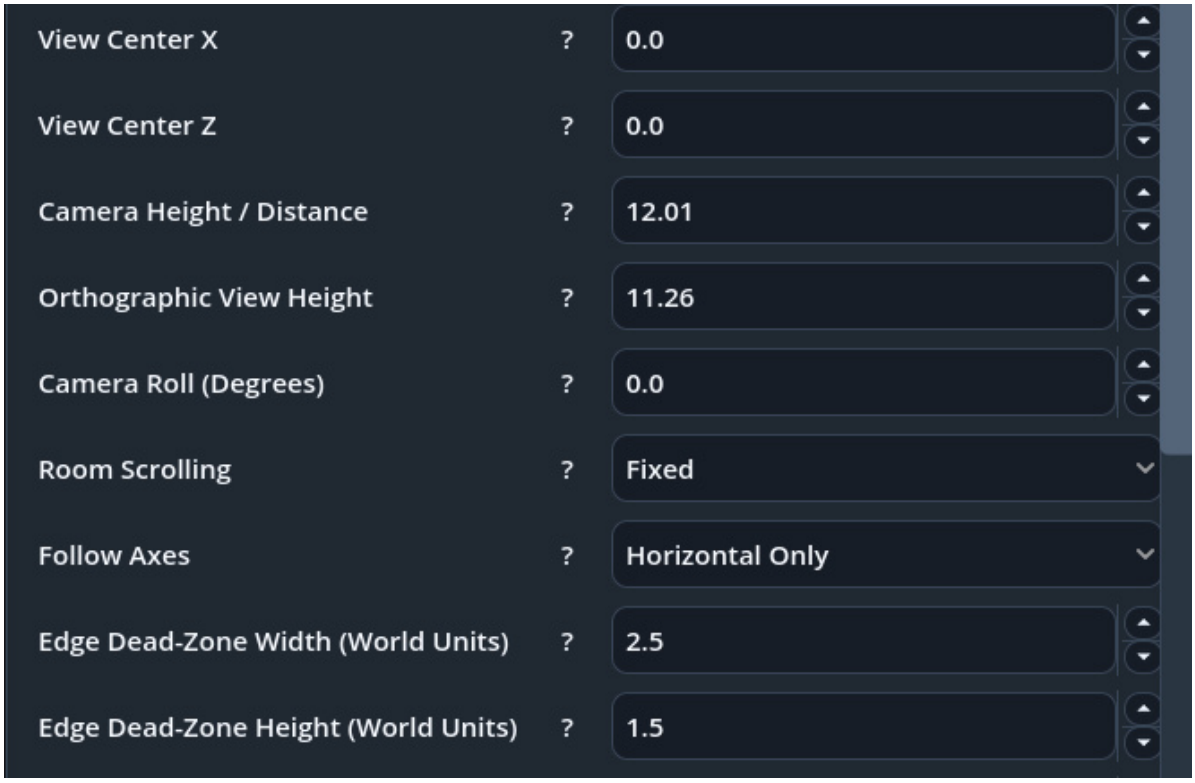

Figure 30. Room Scrolling, Follow Axes and bounds are on the stage’s 2D Camera tab.

1. Open 2D Camera. Begin with an Orthographic View Height such as 11.25.
2. Choose Follow Player for continuous tracking, or Follow Near Screen Edge for a dead-zone approach.
3. Choose Horizontal Only for a side-to-side workshop pan.
4. Set stage bounds to the intended painted area, for example X -16 to 16 and Z -9 to 9 for a centered 32-by-18 background.
5. Enable Smooth Camera Follow and start Camera Follow Speed around 5.
6. For edge follow, choose a comfortable dead-zone width in world units. Test the player near both ends of the scene.
7. Apply and save. Confirm ordinary scrolling before tuning any parallax factor.

Camera bounds control what the camera can reveal. Navigation controls where the character can walk. Give the player enough walkable range to move the view, while keeping the view inside useful artwork.

> A current 2D Stage has one locked orthographic camera. It can frame, zoom, roll and follow within the stage. It is not a perspective camera that can orbit away from the painted plane.

53. Add and select a Parallax Plane

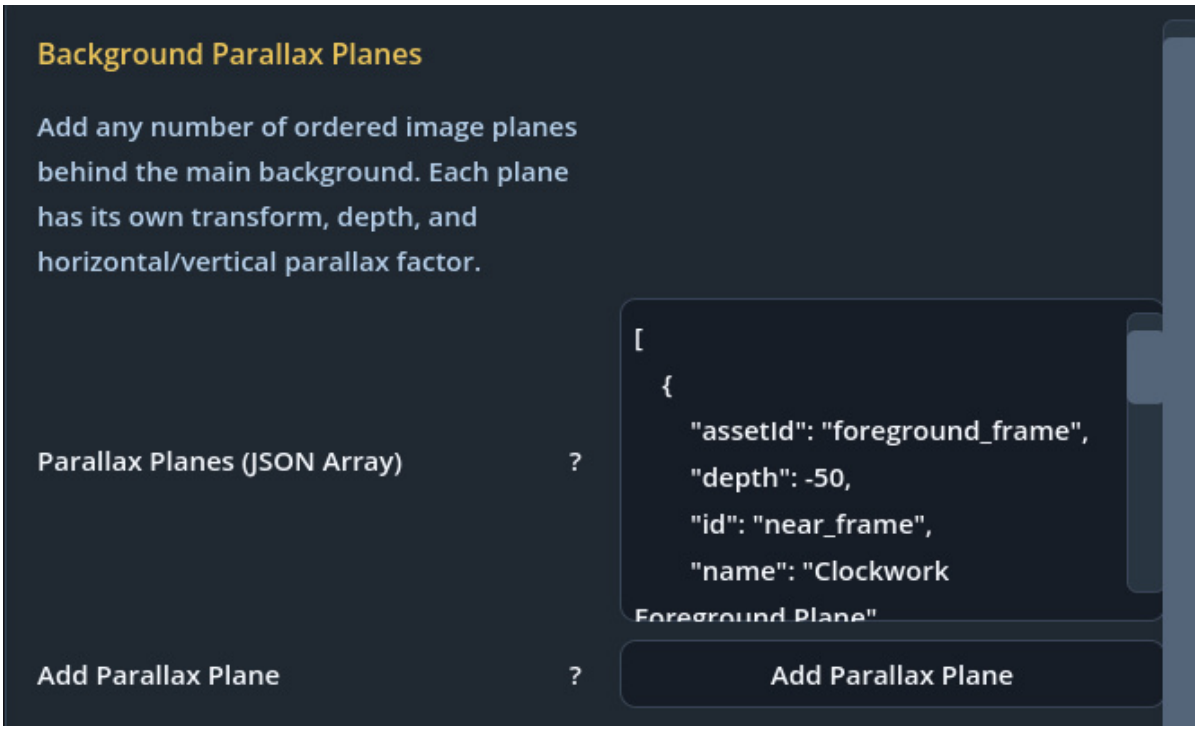

Figure 31. The room’s 2D Layers & Objects tab provides Add Parallax Plane and the saved plane list.

1. Import each painted layer as an Image asset. Use transparency where the scenery behind it should remain visible.
2. Select the room → 2D Layers & Objects → Add Parallax Plane.
3. Select the new plane under Parallax Planes in Scene Objects.
4. Assign its Image Asset and give the plane a clear name and ID.
5. Set Plane Width, Plane Height and Position for the composition. For the sample foreground, use 32 by 18 units centered over the background.
6. Set Layer Order and the horizontal/vertical factors. Apply and test the actual scrolling camera.

You can use a front cutout such as branches or a decorative frame, as well as distant scenery. Check the actual order and transparency: the name “background layer” is not a guarantee that an opaque image will leave other artwork visible.

> The illustrated Clockwork Foreground is a transparent PNG. Its center lets the workshop artwork show through. A solid JPEG in the same place would cover the view.

54. Choose sensible parallax factors

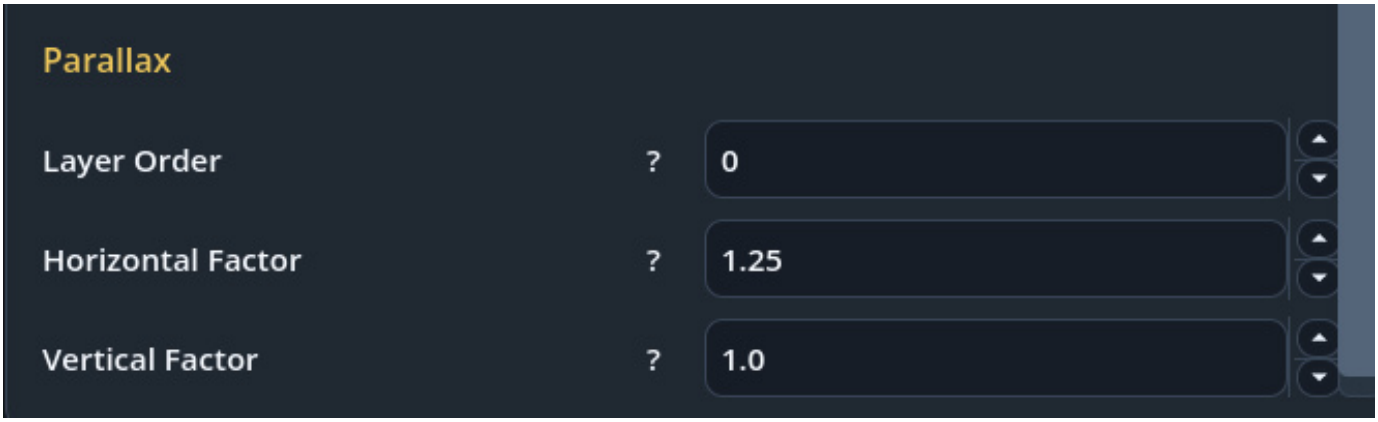

Figure 32. Each selected plane has independent Horizontal Factor and Vertical Factor controls.

Factor	Visible behavior as the camera pans
0	The layer follows the camera enough to remain fixed in the view.
0.25	Small apparent travel; useful for a distant layer.
0.5	Half the apparent travel of a fixed world layer.
1	Ordinary world-fixed artwork. It moves across the screen with the camera pan.
1.25	Faster apparent travel; useful for a nearby foreground cutout.
Negative	Reversed apparent motion. Use deliberately and test carefully.

The two factors correspond to horizontal X and vertical stage Z movement. For a horizontal-only camera, keep the unused vertical factor at 1 unless the composition has a reason to behave differently.

Start with three depths at most: distant 0.25, main artwork 1 and near cutout 1.25. Exaggerated values can make the layers slide apart instead of suggesting depth.

> A factor of 0 does not hide a plane or freeze the camera. It describes that plane’s relationship to camera travel. Visibility is a separate control.

55. Give each layer enough spare image

Parallax can move a layer’s edges into view. A frame that looks perfect at the center can reveal a transparent strip at the left end of the room. Size the artwork for the full camera route, not only the first shot.

Adjustment	What to look for
Widen a near foreground plane	Its faster apparent travel needs extra coverage near the ends.
Oversize a distant sky	A slow layer should still fill every revealed part of the view.
Preserve transparent cutouts	Transparent areas should reveal another intended layer, not empty space.
Check top and bottom	Vertical follow, zoom and camera roll can expose corners even when horizontal coverage looks fine.
Test extreme camera positions	Walk to both edges, use zoomed views if the game supports them, and inspect the final output aspect ratio.

1. Stop the camera at the left limit and inspect the whole frame.
2. Repeat at the right limit.
3. Adjust the layer’s width, position or factor if an edge appears.
4. Repeat after changing camera size or final game resolution.

The current stage planes do not automatically tile a finite image forever. A high parallax factor does not invent extra painted scenery. Use sufficiently wide art or a deliberately designed repeated-layer setup.

56. Parallax is not a walk-behind mask

A parallax plane creates relative image movement. A walk-behind controls when part or all of a character is hidden by foreground scenery. A screen overlay decorates the view above the world. These are three different jobs.

Need	Use
Distant scenery slides slowly	A Parallax Plane with a small factor.
A nearby branch moves faster	A transparent near plane with a larger factor and appropriate composition.
The player walks behind a painted counter	A Walk-Behind image/polygon and an authored baseline.
A fixed decorative frame around the game	Screen Overlay with parallax off.
Clickable frame controls	A GUI with buttons or images and event assignments.

For a counter, a walk-behind baseline decides when the player is behind it. Do not try to solve that by changing only a parallax factor. In a current stage, walk-behind image planes and polygons have their own setup and transparency behavior.

For a 3D prop in a 2.5D scene, use the ordinary scene placement and collision controls. Its visual position, physical collider and interaction volume are separate from image-layer scroll.

> If the character vanishes unexpectedly, inspect walk-behind visibility/baselines and layer order before changing its model or animation. If the camera will not reveal the rest of the picture, inspect camera bounds separately from navigation.

57. Current stages and older native 2D rooms

Room system	Controls and units
Current 2D / 2.5D Stage	Scene Objects → Parallax Planes, world-space X/Z factors, one locked orthographic stage camera. Records are in stage2D.parallaxLayers.
Older native Canvas 2D room	Background/foreground layers in the hybrid/native compositor, pixel-space X/Y and layer ratios. Records use the native layers collection.
Screen Overlay	A single oversized image over the view, driven by the 3D camera target and room navigation bounds. It has Travel Strength rather than one independent factor per plane.

The shared Set 2D Layer Parallax action targets the older native layer system. It is not the current-stage Parallax Plane editor. A layer ID visible in a current stage therefore should not be assumed to work with that legacy action.

```
# For an older native 2D layer, not a Stage Parallax Plane:
await Set2DParallax({"sceneId":"old_room",
    "layerId":"far_sky","ratio":[0.25,1.0]})
```

For a current stage, author the factors in the plane Inspector. A script can edit the stage’s saved configuration, but changing that configuration does not automatically rebuild an already-running set of stage planes. Re-enter or restart the room to test the authored setup.

> This distinction is especially useful when opening projects created by older PCA versions. Identify the room system before following a layer action example from an older manual.

58. Test the parallax as a moving composition

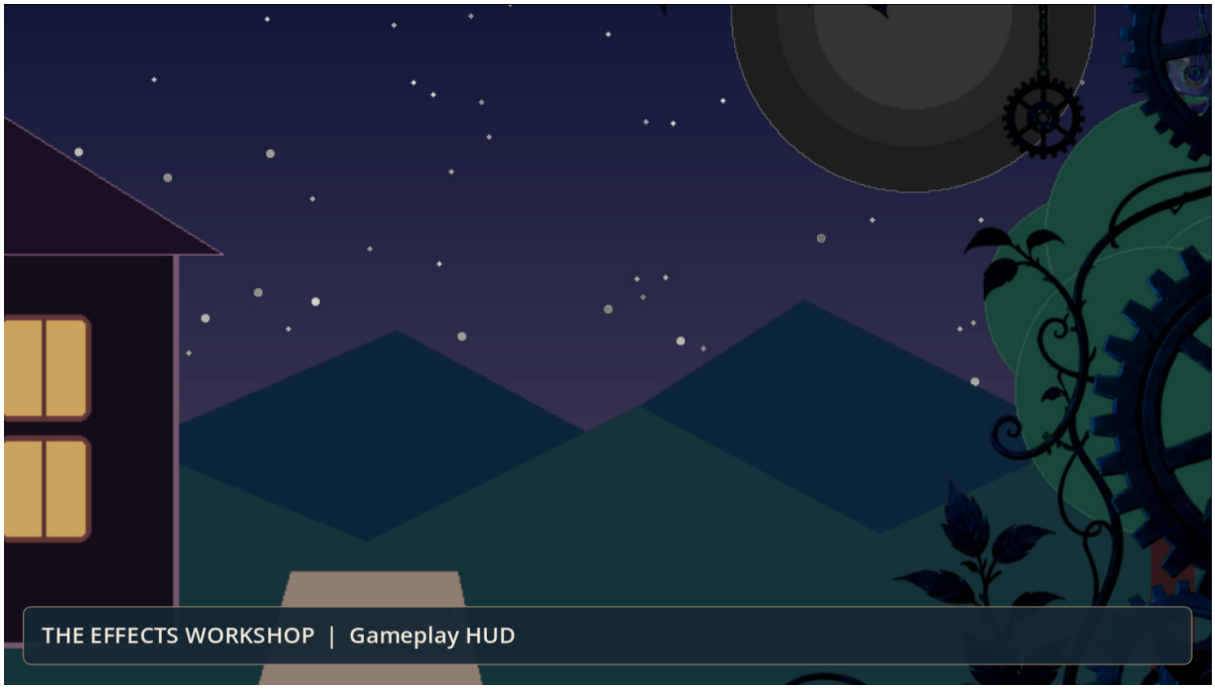

Figure 33. The painted workshop and transparent foreground plane after a horizontal stage-camera move.

1. Start with the camera centered. Confirm which artwork is the main room and which is a separate cutout.
2. Pan or walk toward the left side. Watch a near edge against a recognizable object in the main background.
3. Move right. The relative change should be smooth and predictable.
4. Check each plane’s visibility and image coverage at the limits.
5. Test a dialogue or inventory GUI over the scene. The GUI should not drift with a world layer unless you deliberately built it that way.
6. Save, build and repeat the route in the intended output size.

The guide’s stage example has a main background and a transparent foreground plane with factor [1.25,1]. A camera move of +4 world units moves that plane’s world origin by -1 along X, producing faster apparent travel relative to the main picture.

> The screenshot is a real stage-camera view. It is not proof that a fixed camera will scroll on its own. Choose a follow mode and give the player enough navigation range for the finished game.

59. Screen overlays are decorative image layers



Figure 34. The transparent Clockwork Foreground sits over the world image while the gameplay HUD remains above it.

A Screen Overlay is a single image over the game view. Use it for a decorative frame, a mask, dirt on a lens or other non-interactive artwork that belongs to the view rather than to one object in the room.

Overlay behavior	Practical consequence
Draws above the world	Its opaque pixels can cover scenery. Leave useful transparent areas.
Draws below gameplay GUIs	The HUD, dialogue and authored interface remain available above it.
Ignores mouse input	It is not a clickable hotspot or button.
Uses an image asset	Use the Video GUI element or FMV tools for a movie instead of assuming an overlay image selector plays video.
Can use its own shader source	The frame can remain clean, share room shaders or have its own treatment.

Use a GUI when you need several independently controlled overlay images, buttons, text or animation. The Screen Overlay setting is a simpler decorative layer.

60. Prepare artwork for a screen frame

Create the frame at the intended aspect ratio, with transparency in the part of the view where the game should show through. PNG is a convenient choice. Do not paint the center black unless you want to cover the game with black.

1. Draw the visible edges and decorative details on a transparent canvas.
2. Keep the center clear and avoid covering the usual subtitle or inventory area.
3. Export a transparent image and import it through Asset Designer as an Image asset.
4. Preview the image over contrasting backgrounds if possible. Dark detail can disappear into a dark room.
5. Assign it as an overlay and test the real game view, including the narrowest output aspect ratio you plan to support.

Static frame	Camera-scrolling foreground
Match the design aspect ratio. With parallax off, the image is stretched to the overlay area.	Provide extra image dimensions beyond the overlay design view. Parallax needs spare pixels to crop and move.
Place critical decoration inside a safe margin.	Make the corners and edges look acceptable throughout the camera route.
The artwork itself can be a complete border.	A complete border may be partly outside the view while the image pans. That can be intentional.

The sample Clockwork Foreground is 1672 by 941 pixels. In a 1280-by-720 overlay design area it has spare width and height for travel. A 1280-by-720 image has no spare travel at that same design size.

61. Set a default overlay for the game



Figure 35. Global Settings → Screen Overlay holds the game-wide default image and shader source.

1. Open Global Settings → Screen Overlay.
2. Enable Default Screen Overlay.
3. Choose Overlay Image by its name and ID.
4. Choose No Shaders initially so you can judge the image itself.
5. Leave Parallax Scroll with Camera off for a static frame.
6. Choose whether Screen Transitions Affect Overlay.
7. Apply Changes, save and restart the game preview. Visit a room that inherits the global overlay.

The overlay setting has its own inheritance rules. A room that uses a Custom overlay or disables overlays does not follow this default. This is separate from the room’s Scene Shader Source choice.

> The default overlay applies across rooms that inherit it. Use the next lesson for a frame that belongs only to a telescope view, dream scene or one particular location.

62. Give a room its own overlay



Figure 36. The room can inherit, disable or replace the default Screen Overlay.

Room Overlay Mode	Result
Use Global Overlay Settings	Uses the default overlay from Global Settings.
Disable Overlay in This Room	Shows no Screen Overlay in this room, even if the global setting is enabled.
Use Custom Overlay in This Room	Uses this room’s image and overlay configuration. Choose an image as part of the setup.

1. Select the room → Screen Overlay.
2. Choose Use Custom Overlay in This Room.
3. Assign Room Overlay Image and choose its shader source.
4. Choose its transition and parallax options.
5. Apply, save and use Test Scene. Test another room to confirm that the custom frame is local to this room.

Custom is its own enabled overlay configuration; there is not a second room-level enable checkbox to hunt for. To remove the overlay from that room, choose Disable Overlay in This Room.

63. Choose how the overlay is shaded

Overlay Shader Source	Use it when…
No Shaders	The frame or lens artwork should retain its original appearance.
Use Current Room Shaders	The frame should participate in the room’s chosen visual style. This applies the style to the overlay artwork; it is not a second copy of the world image.
Use Overlay’s Own Shaders	The overlay should have a separate treatment, such as grain, hologram or a subtle tint. Configure its own stack below the source selector.

1. Start with No Shaders and verify the image and transparency.
2. Try Use Current Room Shaders while the room has a clear Sepia or Pixelized treatment.
3. For an independent effect, choose Use Overlay’s Own Shaders and set one layer.
4. Keep an imported overlay shader empty while using built-in layers. If you choose an imported canvas shader, it replaces those layers.
5. Apply and restart the preview after changing the overlay setup.

A canvas surface effect sees the overlay’s image and alpha. Not every specialized world-space feature makes sense on a flat overlay. A depth-based fog distance cannot discover real depth inside a painted frame.

> If the overlay seems to ignore the room’s look, first read Overlay Shader Source. No Shaders is designed to ignore it. If it looks too heavily processed, check for a separate overlay stack as well as the room style.

64. Add camera-driven overlay parallax

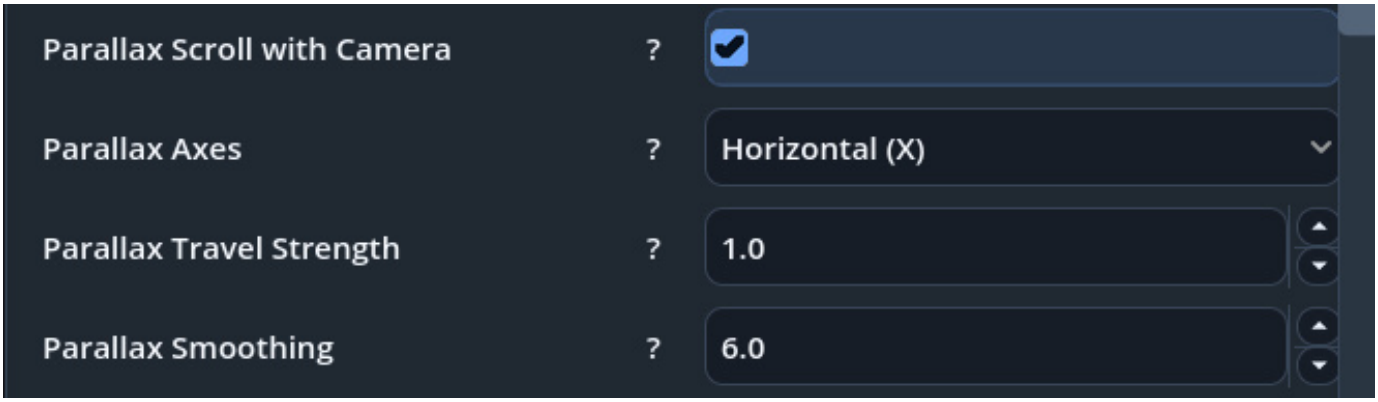

Figure 37. Overlay parallax uses camera travel, spare artwork and the room’s navigation bounds.

1. Use an image larger than the overlay design area in the direction you want to scroll.
2. Enable Parallax Scroll with Camera.
3. Choose Horizontal (X), Vertical / Depth (Z), or both.
4. Start Parallax Travel Strength at 1 and Parallax Smoothing at 6.
5. Confirm valid room navigation bounds. The camera target’s progress across those X/Z bounds drives the image travel.
6. Move the camera target left and right in Test Scene. For an ordinary follow camera, make the player walk far enough to move that target.
7. Check the extreme views and reduce strength if the decoration moves too aggressively.

The image is scaled enough to cover the design area, then cropped and moved within its spare dimensions. If there is no excess width, horizontal parallax has no travel. Changing camera rotation in place without changing its target across the bounds is also not the same as a target pan.

> This is camera-target parallax, not a mouse-follow lens. Smoothing is a response speed: a larger number catches up faster; a smaller positive number gives more lag. Strength 0 keeps the oversized image centered.

65. Overlay size, transitions and live changes

Setting / situation	What to check
Viewport / Overlay Size Mode	The overlay design area can differ from the world rendering size. Compare the assigned image against the resolved overlay area.
Transitions Affect Overlay on	The room-entry transition includes the frame.
Transitions Affect Overlay off	The frame can remain visually stable while the transition changes the room beneath it.
Changing overlay configuration in a script	The saved runtime record can change without rebuilding the current overlay immediately. Verify on room reload/re-entry.
A frame must appear/disappear immediately on an event	Use an authored GUI image/panel and Show GUI / Hide GUI for a direct live lifecycle.

Do not assume every configuration field has a dedicated instant redraw action. In this version, the Screen Overlay is built when the room presentation is built. Its parallax then updates continuously from the camera target, but changing an arbitrary overlay asset property is a separate configuration edit.

1. Test a static frame first, then its camera-driven parallax.
2. Check a cut and a visible room-entry transition with the chosen transition checkbox.
3. Open dialogue, inventory and the options menu over it.
4. If it needs its own event-driven animation or visibility, build that particular visual as a GUI instead.

> Screen overlays do not add interaction. A transparent pixel passes the view through visually; the overlay as a whole already ignores mouse input. Interactive regions belong in the scene or GUI tools.

66. Put the effect on an event

Once an effect is authored, an event decides when it is used. The shared Action List catalog is available in scene events, verb interactions, GUI events, visual scripts and cutscenes. Start with a simple test button before connecting the effect to a complicated puzzle.

1. Create an authored GUI button named Show Hand Effect.
2. Open its Events tab and choose an Action List for Click.
3. Add Attach Particle Effect and select the existing hand_orb emitter and player target.
4. Enter right_hand_grip and the desired offset.
5. Add Change Object Property for hand_orb → particle.enabled → true.
6. Create a second button that disables/hides the effect and detaches it.
7. Test the buttons during gameplay. GUI Preview is useful for appearance; it deliberately does not execute the full game event flow.

Value type	Enter it as
Boolean	true or false; not a text string saying "true".
Number	A numeric value such as 0.4, not a record ID.
Vector	A numeric list such as [0,0.04,0].
Record reference	The actual ID selected by the field, even when the menu shows Name(ID).
Whole shader configuration	An object/dictionary with layers and the relevant option fields.

Additional Event Actions run after a GUI event’s selected primary behavior. If a button already changes a verb, decide whether the effect should accompany that behavior or replace it.

67. Build the same flow with visual nodes

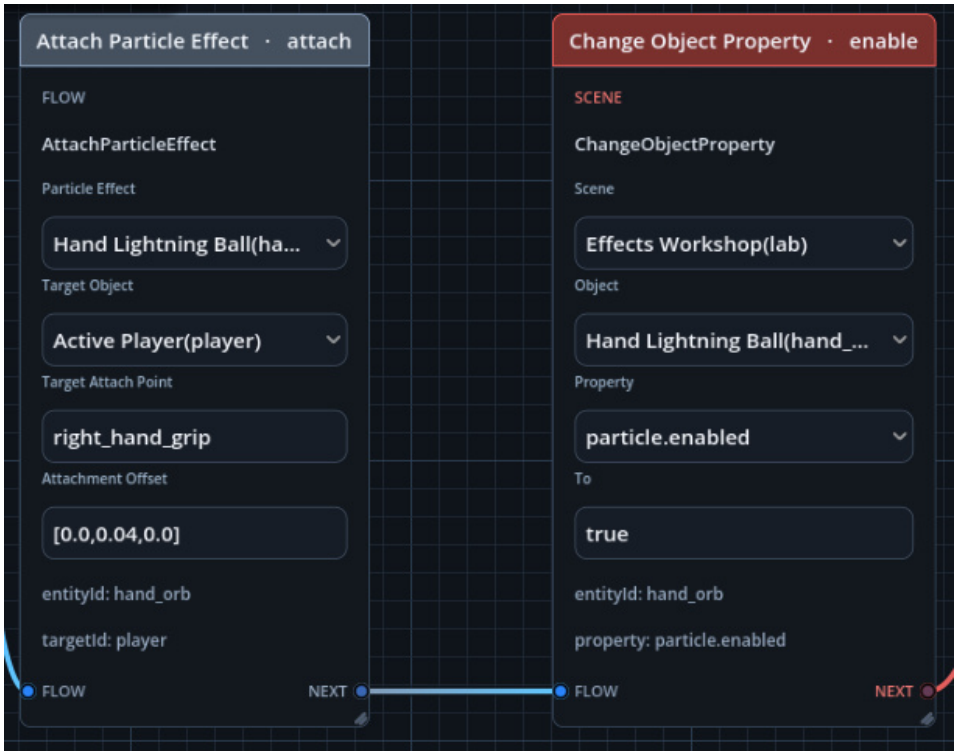

Figure 38. The two action nodes attach the emitter and enable it. The surrounding flow connects them to the entry and End nodes.

1. Create a visual script named Enable Hand Effect.
2. Add the appropriate start/interaction entry node for how you will call it.
3. Add Attach Particle Effect and fill in hand_orb, player and right_hand_grip.
4. Connect its output to Change Object Property. Set particle.enabled to true on hand_orb.
5. Connect the last action to End. Call the visual script from your chosen event.
6. For the stop flow, disable/hide the emitter and then use Detach Particle Effect.

A sequence of visual nodes describes order. It does not mean every started effect has finished before the next node. Attach begins a persistent link; enabling a particle begins emission; Play Sound starts audio. Use an action’s own wait option or a deliberate wait when the story needs a pause.

> Cutscene events use scheduled start times. A waiting action inside one event does not push every later timeline event back. Plan effect starts and cleanup explicitly rather than relying on the width of a decorative timeline block.

68. A complete script to enable the hand effect

Create an enabled GDScript record named Enable Hand Effect, ID effect_on. Keep it non-global with no automatic event subscriptions, then call it from the button or interaction you choose.

```
extends PCAScriptBase

func run() -> void:
    var linked: bool = await AttachParticleEffect({
        "entityId":"hand_orb", "targetId":"player",
        "targetPoint":"right_hand_grip", "offset":[0,0.04,0]
    })
    if not linked:
        return
    await SetObjectVisible({"entityId":"hand_orb","visible":true})
    await ChangeObjectProperty({"entityId":"hand_orb",
        "property":"particle.enabled","value":false})
    await ChangeObjectProperty({"entityId":"hand_orb",
        "property":"particle.enabled","value":true})
```

The script first checks that the attachment succeeded. It then makes the source visible and restarts its emission. This handles an effect that was previously hidden or used once already.

Replace the IDs and point name with your own. The player alias follows the currently active player; an explicit ada target always requests that live instance. The example requires the emitter and target to exist in the same loaded room.

> This is project script code using PCAScriptBase. It belongs in a script record, not in the engine source or a shader source file.

69. A complete script to stop and release it

```
extends PCAScriptBase

func run() -> void:
    await ChangeObjectProperty({"entityId":"hand_orb",
        "property":"particle.enabled","value":false})
    await SetObjectVisible({"entityId":"hand_orb","visible":false})
    await DetachParticleEffect({"entityId":"hand_orb"})
```

Create this as a separate enabled GDScript record, ID effect_off. Assign it to the stop button or ability-ending event. Hiding the ball gives an immediate visual ending; disabling emission and detaching release its active behavior.

If you want…	Adapt the ending
The aura to disappear immediately	Disable emission, hide the source and detach when no link is needed.
A smoke trail to fade naturally	Stop new emission and keep the emitter visible until the old particles have expired.
The effect to remain attached but inactive	Disable emission and keep the attachment. Re-enable it later.
A dropped effect to remain in the room	Detach at its current pose and choose whether it should keep emitting.

Do not copy the hide step into every effect without thinking about the desired tail. A tiny hand aura and a long smoke plume have different useful endings.

> The example’s enable script makes the emitter visible again before restarting it. A visible=false source remains hidden even when you change its particle emission back to true.

70. Switch the inherited screen look by script

Change Game Setting can replace the whole global scene-shader configuration. This is a convenient way to keep a style’s layers and dedicated options together. The room must use the global stack for that change to affect its appearance.

```
extends PCAScriptBase

func run() -> void:
    await ChangeGameSetting({"setting":"globalSceneShaders",
        "value":{
            "layers":[{"preset":"psx","intensity":1.0}],
            "psxPixelSize":3, "psxColorBits":5,
            "psxDither":1.0, "psxJitter":0.0
        }})
```

To clear that runtime look and any held pulse/shake, use this separate record:

```
extends PCAScriptBase

func run() -> void:
    await ChangeGameSetting({"setting":"globalSceneShaders",
        "value":{"layers":[],"preset":"none"}})
    await StopZoomBlur({})
    await StopScreenShake({})
```

Change Project Property with a settings.globalSceneShaders path reaches the same project group. For one dedicated option, Change Game Setting can use a path such as globalSceneShaders.psxColorBits.

Clearing a global stack does not override a room using Custom or Disabled. A restore action also needs a deliberate design: restore the previous configuration if the game already had a normal style, rather than assuming “no effects” is always the correct default.

71. Change a room stack or a caller-owned effect

A room-specific shader configuration belongs to the Scene record. In a GDScript record, its registry proxy can set the same fields used by the Inspector. The active room refreshes its environment/post process when those fields change.

```
extends PCAScriptBase

func run() -> void:
    Scene.Get("lab").Set(
        "environment.postProcessOverrideMode", "custom")
    Scene.Get("lab").Set("environment.postProcess", {
        "layers":[{"preset":"sepia","intensity":0.6}],
        "tintStrength":0.0
    })
```

For a caller-owned object, the shared action API also supports this. For example, an event owned by an authored Heat Haze particle can change its own distortion:

```
await ChangeObjectProperty({"entityId":"this",
    "property":"particle.heatDistortion","value":0.0})
```

The caller is captured from the event context. On a GUI event, this identifies the GUI/calling element context, not the brazier. On an object event, it identifies that object. A standalone script test or cutscene test may have no object caller at all.

> Use explicit IDs for cross-object operations. “this” is not a universal substitute for “whatever I last selected in the editor.” It belongs to the running event’s caller.

72. Use the same actions in PCA Script

PCA Script can call the same shared actions. Keep each dictionary expression on a single physical line in this example. The editor’s visual line wrapping is fine; inserting newlines inside the expression changes how this parser reads it.

```
language 1.0
function run()
    let effect = "hand_orb"
    AttachParticleEffect({entityId:effect,targetId:"player",targetPoint:"right_hand_grip"})
    ChangeObjectProperty({entityId:effect,property:"particle.enabled",value:true})
end
```

For simple property changes, PCA Script’s readable property syntax is useful:

```
language 1.0
function run()
    set objects.hand_orb.particle.enabled to false
    set objects.heat_plume.particle.heatDistortion to 0
end
```

Language / tool	Use the shared effect contract
GDScript	await AttachParticleEffect(options) and the typed action helpers.
PCA Script	Shared action calls or readable set objects.<id>.<path> to value statements.
Lua / BASIC	Their existing shared action calls use the same IDs, option names and value types.
Visual scripts / action lists	Choose the catalog action and edit its parameter fields.

The language changes how you write the call; it does not remove rendering limits or make every particle field rebuild live. Keep the authoring/runtime distinctions from lesson 37.

73. Combine a burst, sound and zoom blur



Figure 39. A held Zoom Blur makes its radial effect easy to inspect. Use a short timed pulse for ordinary gameplay.

```
extends PCAScriptBase

func run() -> void:
    await ChangeObjectProperty({"entityId":"blast",
        "property":"particle.enabled","value":false})
    await ChangeObjectProperty({"entityId":"blast",
        "property":"particle.enabled","value":true})
    await PlaySound({"assetId":"chime","channel":"impact"})
    await ZoomBlur({"strength":0.3,"duration":0.6,
        "centerX":0.5,"centerY":0.5,"clearRadius":0.12,
        "includeGUI":false,"wait":false})
```

This starts an authored explosion and a sound, then adds a 0.6-second radial pulse. Clear Center Radius keeps the focal region less blurred. Center X/Y are normalized coordinates: 0.5,0.5 is the center.

A positive Zoom Blur duration rises and falls, then clears. Duration 0 holds the effect until Stop Zoom Blur. A new blur replaces the old one. Keep Include GUI off initially so dialogue and inventory stay readable.

74. Add shake, snow changes and restrained motion

Action	Use
Zoom Blur / Stop Zoom Blur	A radial pulse or a deliberately held state. Strength is 0-1; duration is seconds.
Camera Shake	Offsets a 3D camera. It is distinct from shaking the composed screen.
Shake Screen / Stop Screen Shake	Shakes scene, FMV and UI layers. Power is in pixels; duration is seconds; choose both/horizontal/vertical.
Set Room Tint	Changes the room post-process tint color and strength. It sets a custom room treatment.
Set Snow Accumulation	Changes visual surface accumulation amount with duration or pace. Author the snow treatment first.
Melt Accumulated Snow	Animates the accumulation down. It does not delete ordinary snow particles.

```
await ShakeScreen({"direction":"both",
    "duration":0.3,"power":5.0})
# For an authored Snow screen/surface setup:
await SetSnowAccumulation({"sceneId":"lab","amount":0.8,
    "duration":10.0,"waitForCompletion":false})
```

Use a small shake for a small event. A player should still understand the room and the hotspot position after it. If you expose an effects preference in the game’s GUI, let it reduce strong grain, blur, flicker and shake as well as particle density.

Depth of field is a camera feature requiring Mobile or Forward+. Volumetric fog requires Forward+. In this version ordinary fog, glow, shadows and reflection probes have paths across the three renderers. Restart the editor/game backend when changing renderer; restarting only the preview does not change the editor backend.

75. Make cutscene cleanup explicit

Cutscene skipping does not mean “execute all remaining events instantly.” Future events can be skipped. If a late event was going to hide an aura or stop a sound, an early skip may never reach it.

1. List every temporary effect the sequence starts: particles, attachment links, extra GUIs, sound channels, room tint, held blur and shake.
2. Decide the useful final state for normal completion and for a skip.
3. Keep a cleanup script outside the cutscene timeline. Call it after the cutscene finishes or is skipped.
4. Disable/hide temporary emitters, detach links that should end and stop named sounds.
5. Clear a held Zoom Blur or Screen Shake and restore the intended screen look.
6. Test skipping during the first title, during movement and while a major effect is active.

Temporary change	Deliberate cleanup
Hand aura	Disable/hide it and release the link if the ability ends.
Brazier lit permanently by the story	Keep it lit and save the appropriate story state.
Held blur or tint	Restore the intended normal look.
Extra frame built as a GUI	Hide that GUI.
Looping effect sound	Stop its specific channel rather than every sound in the game.

Automatic camera/HUD restoration is not a complete undo system for every scene edit. A good effect script owns its beginning and its ending.

76. Diagnose a missing or misplaced effect

Symptom	Check in this order
No room shader change	Apply/save; room source mode; imported shader override; correct room; current preview.
Only one object should change but everything changes	You edited a room/global stack. Select the object and use its Visual Effects.
A particle never appears	The actual gameplay camera; emitter position/size; Emit In Game; entity visibility; one-shot already finished.
A custom particle still has the old image	Check Particle Image. Choosing another preset can retain it.
A hand effect sits at the feet	Target ID; blank point field; exact point name; point enabled state.
The effect drifts away during animation	Bone binding, target point rotation, per-clip override and local coordinate choice.
A changed particle count looks unchanged	Check whether that preset/field refreshes live. Restart/re-enter after applying authored settings.
A mirror shows only highlights	Reflection Mode should be Live Mirror for a flat character reflection; inspect axis and view geometry.
Heat looks invisible	Place detail behind it; check source visibility, opacity and distortion; use the right surface/particle property.

Return to one effect, one camera and a simple background when a combination becomes hard to understand. Get that working first, then reintroduce the other layers one at a time.

> Use Project Validation for broken IDs, missing images and script errors. A visually plausible selector value can still refer to a record that was renamed or removed.

77. Diagnose parallax and overlay problems

Symptom	Check
Parallax is still	Move the camera; confirm the factor differs from 1 where appropriate; identify stage versus legacy layer system.
A Stage plane is missing	Image reference, visible state, size, position, stacking and opaque art covering it.
The character stops before the painted edge	Navigation/collider limits. Enlarging the picture or overlay is not a navigation change.
The camera stops too early	Stage camera bounds and visible view size.
Overlay does not move horizontally	Parallax enabled; X axis selected; excess image width; camera target X travel; valid navigation bounds.
Overlay looks stretched	Static overlays stretch to the overlay design area. Match the artwork aspect ratio or use a suitable GUI image fit.
Overlay covers the whole game	Check image alpha. An opaque center covers the scene.
Overlay ignores room effects	Read Overlay Shader Source. No Shaders is an explicit choice.
Runtime overlay property changes appear later	The configuration does not rebuild every current overlay instantly. Test after room presentation reload, or use a GUI for immediate event-driven visuals.

Compare the final game resolution with the editor view. Camera size, render viewport size, GUI design size and overlay size can be related without being identical. Always test the mode that the player will actually use.

> Changing a layer’s visual position should not be your first attempt to fix an unrelated interaction or collision problem. Keep the image, navigation and collider jobs separate.

78. Finish and test the complete scene

1. Apply every edited asset, point, room and object record. Save Project.
2. Run Project Validation and fix missing references or script errors.
3. Restart Test Scene and watch the first seconds of every one-shot effect.
4. Move the player, play gestures and inspect attached effects from more than one camera angle.
5. Run the real gameplay route that turns the effect on and off. Repeat it.
6. Test cutscene skipping, room changes and save/load when those routes use the effect.
7. Inspect both extremes of every scrolling view, then open dialogue, inventory and options over the final composition.
8. Compare renderer diagnostics with the intended game renderer. Check the slowest target device.
9. Build the game, keep its complete output together and run it outside the editor.
10. Check credits and licenses for imported shaders, images and sounds before distributing.

Use the example room as a reference for scale and control relationships, not as a requirement that every game use the same look. A quiet room may need a little dust and a warm light; a magical machine may justify a bright bolt and a short pulse.

> The best final check is a normal playthrough. The player should notice the important effect, understand what happened and still be able to read and use the game.

Appendix A. Particle property reference

Use these paths with Change Object Property for the authored Particle entity. Values and visual refresh behavior depend on the selected preset; see lesson 37 before planning a live change.

Property	Value / purpose
particle.enabled	Boolean: starts/stops emission.
particle.oneShot	Boolean: one emission cycle.
particle.localCoords	Boolean: whether ordinary particles travel with the source.
particle.amount / particle.lifetime	Base count / seconds of life.
particle.preprocess	Seconds of initial simulation for a looping setup.
particle.size / particle.opacity	Base piece size / visibility 0-1.
particle.imageAssetId	Image asset ID, or an empty string for default artwork.
particle.velocityMin / velocityMax	Use the particle. prefix on each; starting speed range.
particle.direction / particle.gravity	Three-number arrays for direction/acceleration.
particle.areaExtents	Three-number source-volume extents.
particle.spread	Angular spread.
particle.randomness / explosiveness	Use particle. on both; timing variability / burst concentration.
particle.color / emissionColor	Use particle. on both; color strings such as #80bfff.
particle.turbulenceEnabled	Boolean.
particle.turbulenceStrength / turbulenceScale	Use particle. on both; irregular-motion controls.
particle.heatDistortion / heatSpeed	Use particle. on both; heat refraction strength / pattern speed.

Some table rows shorten a repeated prefix for readability. For example, the full second path beside particle.velocityMin is particle.velocityMax. Do not paste the slash-separated table text as a single property.

A2. Bolt and attachment references

Lightning property	Purpose
particle.boltWidth / boltLength	Use particle. on each; thickness / straight-bolt length.
particle.flashDuration / flashInterval	Use particle. on each; visible time / dark interval, in seconds.
particle.boltJaggedness / boltSegments	Use particle. on each; irregularity / segment count.
particle.ribbon	Boolean: camera-facing flat arc/bolt geometry.
Lightning Ball: particle.size	Diameter; straight Bolt Length is not used.

Attach Particle Effect option	Example
entityId	hand_orb: an already-authored Particle entity.
targetId	player or a live entity ID such as ada.
targetPoint	right_hand_grip; empty means target origin.
offset	[0,0.04,0], relative to the point.

The Inspector stores the initial particle link as attachmentTargetId, attachmentPoint and attachmentOffset. For a live link change, use Attach Particle Effect or Detach Particle Effect so the runtime attachment relationship is updated.

> Persistent attachment operates within the loaded room. Self-links, cycles, unavailable named points and incompatible native 2D/3D pairs are rejected. A missing point should be corrected, not silently replaced with a guessed room position.

Appendix B. A complete screen-style value

This is the value object for Change Game Setting → globalSceneShaders. It is not a whole project file. Rooms using the inherited stack receive it; Custom and Disabled room choices continue to override it.

```
{
  "layers": [
    {"preset":"psx", "intensity":1.0},
    {"preset":"horror-fog", "intensity":0.65}
  ],
  "psxPixelSize":3,
  "psxColorBits":5,
  "psxDither":1.0,
  "psxJitter":0.0,
  "horrorFogAmount":0.65,
  "horrorFogStart":3.0,
  "horrorFogEnd":13.0,
  "horrorFogColor":"#202b35",
  "horrorFogGrain":0.02,
  "horrorFogVignette":0.25,
  "tintStrength":0.0
}
```

A simple off value is:

```
{"layers":[], "preset":"none"}
```

For a room-specific value, the corresponding configuration is environment.postProcess on that room, with environment.postProcessOverrideMode set to custom. Use the record proxy example in lesson 71 for the active room refresh.

> If an imported shaderAssetId is assigned, that custom shader takes precedence over the built-in layers. Clear it deliberately when returning to built-in presets.

Appendix C. A current-stage parallax plane

This object describes one current Stage Parallax Plane. It belongs inside that room’s stage2D.parallaxLayers array, not in the project root or an older native layer array. Prefer the visual Inspector while learning.

```
{
  "id":"near_frame",
  "name":"Clockwork Foreground Plane",
  "assetId":"foreground_frame",
  "visible":true,
  "order":0,
  "parallax":[1.25,1.0],
  "worldSize":[32.0,18.0],
  "position":[0.0,-0.06,0.0],
  "rotation":[0.0,0.0,0.0],
  "scale":[1.0,1.0,1.0],
  "depth":-50,
  "pixelPerfect":false
}
```

Field	Interpretation
assetId	A separately imported image with useful transparency.
worldSize	Width and height across the stage’s X/Z plane.
parallax	Horizontal and vertical factors. 1 is world-fixed; a larger factor moves faster in view.
position / rotation / scale	The layer’s own authored transform.
visible / order / depth	Visibility and authored composition controls. Test the actual layer stacking.

A configuration edit does not promise an immediate rebuild of existing stage planes. Save and test a fresh room presentation. Do not use Set 2D Layer Parallax as though it were a current-stage plane setter.

Appendix D. Overlay configuration reference

The global configuration is settings.screenOverlay. A room also has screenOverlay with overrideMode. These are separate from GUI image elements.

```
{
  "enabled":true,
  "assetId":"foreground_frame",
  "shaderMode":"none",
  "transitionAffectsOverlay":true,
  "parallaxEnabled":true,
  "parallaxAxes":"x",
  "parallaxStrength":1.0,
  "parallaxSmoothing":6.0,
  "shader":{"layers":[],"preset":"none"}
}
```

Option	Allowed meaning
shaderMode	none, room or own.
parallaxAxes	x, z or xz. Z progress drives vertical image travel.
parallaxStrength	0-4; travel is limited to the available excess image.
parallaxSmoothing	Positive response speed; higher catches up faster.
Room overrideMode	inherit, disabled or custom.
Image resolution	Excess dimensions are required for travel at the resolved overlay design size.

The current overlay reads an image texture. Its camera-driven travel is continuous once built. Arbitrary configuration changes need the room presentation to rebuild; use a GUI when immediate scripted image swaps and independent animation are required.

Appendix E. Renderer and setup reference

Feature	Current backend rule in PCA 0.86.3.4
Ordinary scene fog, glow, shadows and reflection probe	Available paths across Compatibility, Mobile and Forward+.
Camera depth of field	Mobile or Forward+; also limited by transparent layered presentation.
Volumetric fog / SDFGI / screen-space reflections	Forward+ paths.
Standard world particles	Check the selected renderer; turbulence, refraction and transparency appearance can differ.
Imported shaders	The source must match its canvas/spatial destination and the selected renderer.
Editor backend change	Requires restarting PCA; Play / Restart restarts the game preview only.

Use the preview renderer information and rendering diagnostics rather than assuming the project’s selected game renderer is already the running editor renderer. Keep the target device in the final test plan.

The screenshots here were captured with the Compatibility renderer. No native Windows, macOS, Android or iOS device performance result is implied by the screenshots.

> A shader-heavy room should still be understandable with a simpler look. Keep puzzle-critical color, contrast and interaction cues readable under the final treatment.

Appendix F. Credits for shader work

PCA 3D Adventure Studio is created by Joseph DiPerla. The following shader credits are reproduced in condensed form from the bundled Credits information. Keep the supplied attribution and license files with redistributed components.

Effect / contribution	Credited creator
PSX shader / console integration reference	Francisco Cidade; CC0 source/adaptation.
Aurora	BluePhoenix1.
Frostbite	TTien63.
Broken Glass	miwls.
Wind Sway	kaeylos.
Metallic Toon	ProfesorShader.
Screen Fire	dardevv.
Photography	TuniTem.
Hand Drawn	Br0skinator.
Classic Dithering	zessbin.
Glass Pixel	BuldiDev.
Hologram	binbun.
Pixelized / Pixelated	aricchi; MIT source.

The listed adapted artistic sources are identified as CC0 in the bundled credits except where noted. See the engine’s Credits view and supplied CREDITS.md for source works and complete notices.

> Credits identify provenance. They are not permission to copy unrelated images, music or paid shader packs into a game. Check the license of any additional asset you import.

F2. References and independent implementations

Reference / component	Credit and distinction
VHS / CRT 2 and Film Grain visual references	ekkochambers. The bundled credits identify these as reference-only; PCA uses independent implementations rather than embedding the GPL shader sources.
Zoom Blur visual inspiration	Vefects / Easy Zoom Blur. The restricted pack is not bundled; PCA’s Zoom Blur is independently implemented.
Low Poly conceptual reference	Godot Forum contributors; the effect is not a polygon-reduction importer.
Screen Flicker and original PCA work	Joseph DiPerla / PCA3D.
Underlying engine	Godot Engine.

The guide uses the engine’s existing workshop illustration, foreground frame, dust artwork and bundled Clockmaker models for editor demonstrations. It does not include a separate asset pack or practice project.

When using a third-party shader, preserve its supplied notice and any required attribution. The engine’s own adaptation notes are the appropriate starting point for the included components; your additional imports can have different terms.

> This tutorial explains the available controls and workflows. It does not change the engine’s code or add a new shader integration.

Version and verification notes

Prepared against PCA 3D Adventure Studio 0.86.3.4, its current Inspector and native editor/runtime. All editor and Play figures are actual captures, cropped only for the page composition.

The practical checks cover room-wide PSX/N64/Horror Fog settings, a hand-bound Lightning Ball, movement while attached, missing-point rejection, stopping/detaching, ordinary local/world-coordinate particle settings, explosion re-triggering, a held Zoom Blur, and the illustrated scripts.

The overlay example is tested as both a static frame and an oversized camera-target parallax image. The current-stage example is checked with a real orthographic camera move and the plane’s resulting position.

Controls and limits are checked against the bundled editor/runtime implementation, including shader inheritance, imported-shader precedence, particle live-refresh behavior and the distinction between current stages and older native 2D layer actions.

No engine source changes or separate practice files are required. Adapt the names, IDs, dimensions, offsets and final cleanup to the project you are building.

> The appendices are reading/copy references inside this PDF. They are not additional downloads. Use the clickable contents and PDF bookmarks to return to any lesson.