PCA 3D ADVENTURE STUDIO

Global Settings & Project Settings

Set up the game your players will use



Choose the defaults. Understand the overrides.

Startup, resolution, rendering, movement, interfaces, player options, saving, translations and distribution names. A practical guide to both settings designers, with actual editor screenshots and a working runtime example.

Version 0.86.3.4 - Windows, macOS and Linux

No practice project or extra asset pack required.

How to use this guide

Set up a straightforward game first, then add the presentation and device features you need. These designers cover a large area; you do not have to enable every system to finish a good adventure.

Chapters	What they cover
1-2	Which designer to use, the actual page layout, applying, saving and testing.
3-8	Identity, sounds, resolution, startup, interaction, splashes, packaging and translations.
9-12	Renderer, quality, grading, lighting, viewport presentation and hover names.
13-18	Movement, controls, cursors, dialogue, script editing and camera following.
19-23	Scene effects, overlays, system GUIs, player options and saving.
24-26	Controllers, touch, scene navigation and VR configuration.
27-28	A working baseline, runtime actions/scripts, a location index and troubleshooting.

The screenshots were captured from the running editor. Close-ups are cropped to make the controls readable. The small workshop and bundled Clockmaker records provide example names in selectors; use the assets and records in your own game.

The capture session uses Compatibility rendering. Screenshots of Forward+ Enhanced or device-specific controls demonstrate their configuration; they do not claim that a different renderer, gamepad, or headset was tested here. The basic startup, runtime options and shared control changes were checked in the running game.

Apply Changes, Save Project, then test. Check a full Play / Restart as well as a Test Scene. A scene override or saved player options can explain why a global default is not the value you see during play.

Contents

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

1. Know which settings window you need . . . 4
2. Open the designers, apply changes, and test . . . 5
3. Set the title, default sounds, and resolution . . . 7
4. Choose the starting player and opening sequence . . . 9
5. Choose interactions, inventory, and 2D defaults . . . 11
6. Build the startup splash sequence . . . 14
7. Name the distribution files and choose the icon . . . 18
8. Add game languages and hand translations to other people . . . 19
9. Choose a renderer and quality level . . . 22
10. Use cinematic grading and recognize the two retro paths . . . 24
11. Configure lighting, reflections, shadows, and antialiasing . . . 27
12. Keep the GUI size steady and show hover names . . . 29
13. Set walking speed and automatic approach . . . 31
14. Choose movement profiles and input switching . . . 32
15. Configure keyboard movement and cursors . . . 35
16. Choose the dialogue presentation and response rules . . . 37
17. Set up the native GDScript editor . . . 39
18. Set shared dynamic-camera following . . . 40
19. Build the global scene-shader stack . . . 42
20. Add a screen overlay and optional parallax . . . 46
21. Assign the system GUIs . . . 48
22. Choose the player's starting options . . . 50
23. Set saving rules, screenshots, and the main menu . . . 53
24. Map gamepads and generic joysticks . . . 55
25. Set touch gestures, first-person sticks, and player camera navigation . . . 57
26. Configure VR defaults . . . 60
27. Make a working baseline and change settings during play . . . 62
28. Find a setting, troubleshoot it, and finish the project . . . 65

1. Know which settings window you need

Project Settings and Global Settings both belong to the game you have open. Their names describe two groups of project data, not two separate games or two competing ways to save the same thing.

Use **Project Settings** to establish the game's identity, resolution, starting character, startup sequence, interaction mode, inventory arrangement, splash screens, and distribution names. Use **Global Settings** for the defaults that the game's rooms and systems share: rendering, movement, controls, interfaces, player options, and saving.

This guide uses PCA 3D Adventure Studio **0.86.3.4**. It applies to Windows, macOS, and Linux. The pictures come from the running editor, using a small workshop scene and bundled Clockmaker records for the selectors. There are no practice files to install.

Three kinds of settings

Kind	Examples	Where it belongs
Authored game defaults	Starting player, interaction mode, renderer, default volume.	Project Settings and Global Settings; saved with the project.
Room, camera, character, or element choices	One room's shaders, a camera's follow behavior, a character's walk speed.	The appropriate designer or Inspector. These may override a shared default.
Editor/device preferences	Editor theme, editor interface language, UI size, panel layout, build-tool paths.	The editor's toolbar/preferences or Build Tools. These are not all game defaults.

Global does not mean “change every project on this computer.” Most controls in that designer are saved in the current game. General also contains a few project-stored editor choices, such as native GDScript formatting. One notable device-specific command is the button that remembers which renderer the editor should use at its next launch. Chapter 9 explains the distinction.

A useful order for a new game

1. Give the game its title and resolution.
2. Choose the starting player and the startup action.
3. Decide how the player moves and interacts.
4. Select a renderer and a sensible quality level.
5. Assign interfaces, player-option defaults, and saving rules.
6. Add presentation extras after the basic game works.

7. Validate, save, and test both a full startup and a room test.

You can return to these choices later. The aim is to make them deliberate, so you are not diagnosing a camera problem that turns out to be an input profile or a room override.

2. Open the designers, apply changes, and test

Use the workspace toolbar's **Project Settings** and **Global Settings** icons. On a desktop, hover over an icon to read its tooltip. The Project page is identified by its project/document icon; Global Settings uses the settings/gear icon.

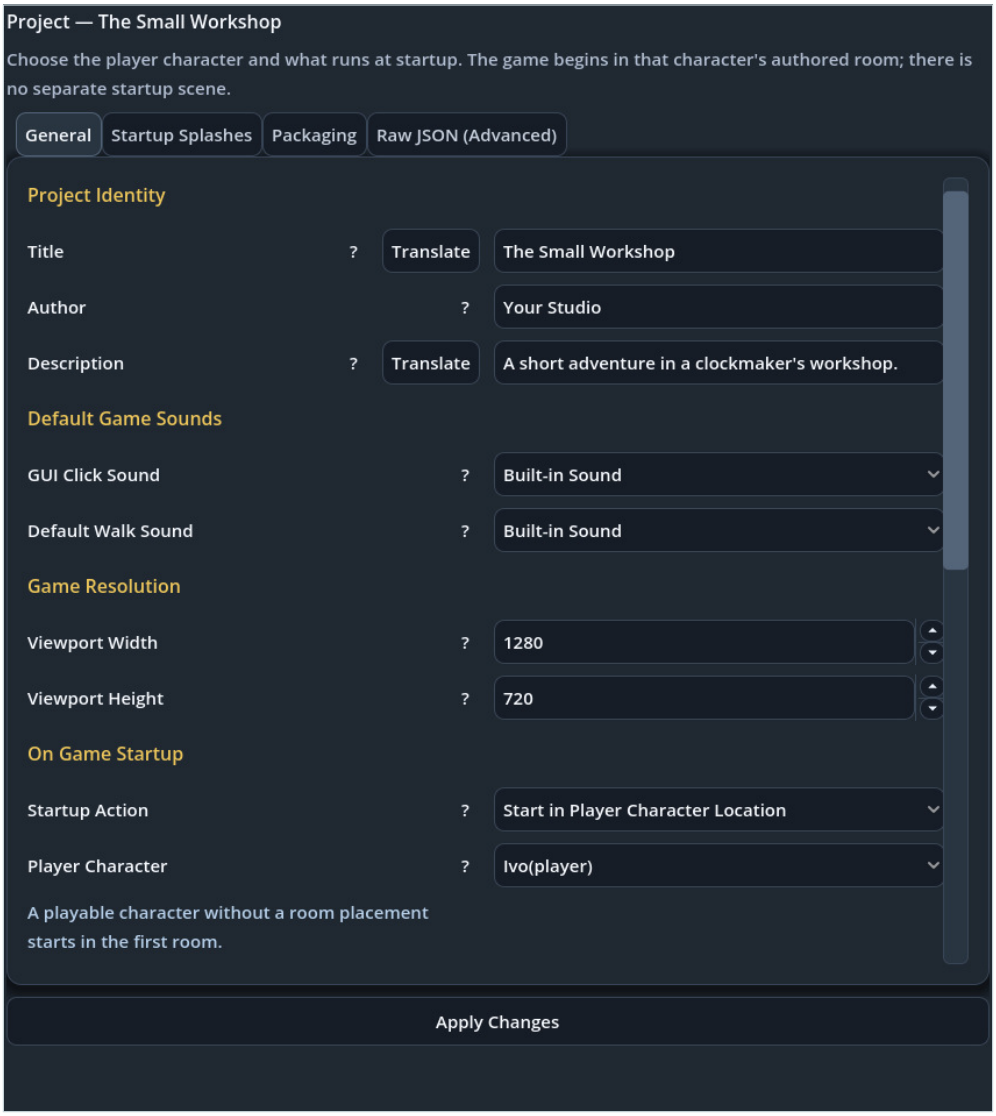

Figure 1. Project Settings has its own editing form and section row. The record title at the top confirms which designer is open.

Work through the actual pages

Project Settings has **General**, **Startup Splashes**, **Packaging**, and **Raw JSON (Advanced)**. Its **Game Languages & Translations...** button opens the game-localization window.

Global Settings has these pages:

Page	Main purpose
General	Rendering, grading, enhanced lighting, presentation defaults, hover names, script-editor controls, global scene shaders, and system-GUI assignments.
Adventure & Cursor	Walking, approach, movement/input profiles, keyboard controls, dialogue behavior, and cursors.
Screen Overlay	The default overlay, its shaders/parallax, Save Slots, and Show Main Menu on Start.
Dynamic Camera	Shared player-follow defaults.
Player Options	Audio/display/accessibility defaults, autosaving, save screenshots, and fonts.
Controllers	Gamepad and generic joystick mappings.
Mobile Controls	Touch gestures and first-person touch sticks.
Scene Navigation	Player camera zoom and pan.
VR Controls	Locomotion, interface placement, and headset-controller mappings.
Raw JSON (Advanced)	The stored settings record.

In this release, **Native GDScript Editor**, **Default Global Scene Shader Stack**, and **Custom System Interfaces** are groups farther down **General**. They are not separate tabs. Use the page's vertical scrollbar to reach them. The small arrows on the section row reveal tabs that do not fit across the window.

Apply is different from Save Project

Apply Changes accepts the values in the form. **Save Project** writes the project to disk. Use both before relying on a change. If a draft prompt appears while you are leaving a page, read it and choose whether to keep or discard those edits.

Try one setting at a time when learning a system. Apply it, test the result, then continue. Changing a renderer, movement profile, dialogue style, and custom GUI together makes a failure much harder to trace.

Play / Restart and Test Scene answer different questions

Play / Restart starts the game through its authored startup settings. Use it for splash screens, startup cutscenes/scripts, the starting player, and the main menu. **Test Scene** isolates a chosen room for quick scene work. It is not a complete test of your game's opening sequence.

A preview restart also gives the game a fresh set of authored defaults. It does not restart the editor process or switch its rendering backend. Stop and restart the whole application when the renderer instructions require that.

3. Set the title, default sounds, and resolution

Open **Project Settings > General**. Begin with the top of the page.

Identity

Title is the game's readable name. **Author** and **Description** are project information. They are separate from the names used for executable files and from the author name entered in any particular asset's credits.

1. Enter a title you can recognize in a test window.
2. Enter the intended studio/author credit and a short description.
3. Apply, then save the project.

The example uses **The Small Workshop**. A title can contain spaces even when a distribution filename should be simpler. Chapter 7 covers package naming.

Default Game Sounds

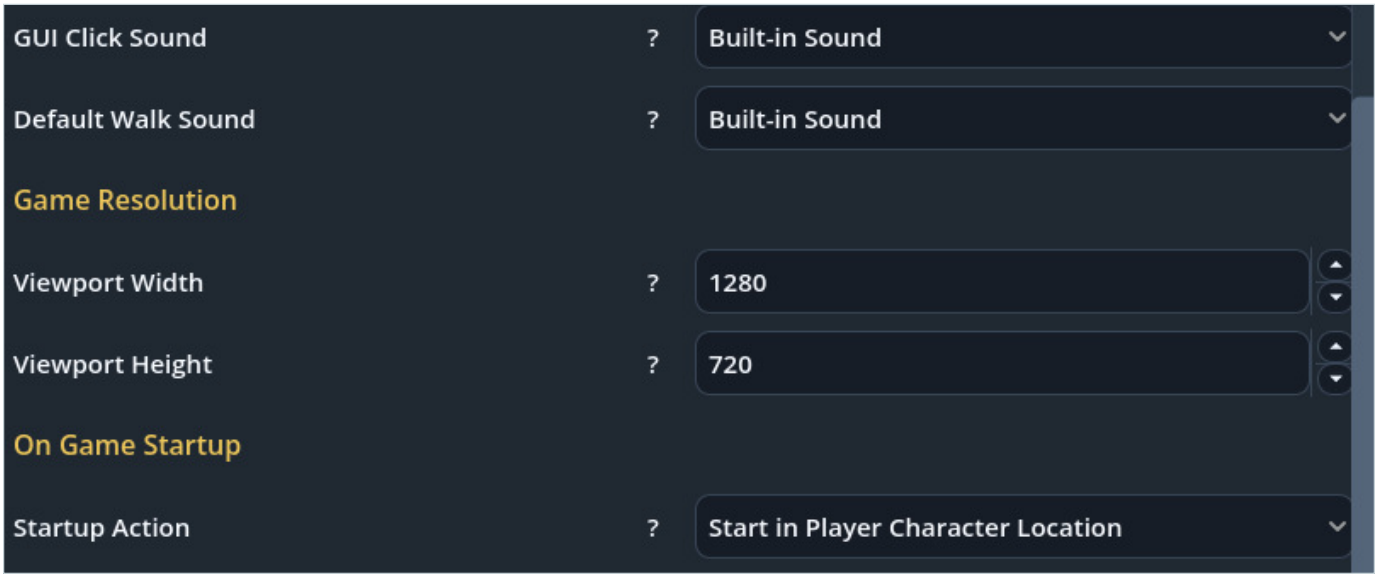

Figure 2. The same part of General contains the default click/footstep sounds and the base viewport dimensions.

Field	What it does
GUI Click Sound	Chooses the default click sound used by the game's GUI controls.
Default Walk Sound	Chooses the shared footstep sound where a character uses the project default.
Built-in Sound	Uses the supplied default rather than an imported Audio asset.
No Sound	Deliberately disables that default sound.

Import a custom sample as an Audio asset first, then select it by its readable name and ID. Keep clicks short. For footsteps, use a suitable individual step rather than a recording of someone walking across an entire room.

A character can choose a **Walk Sound Override** and its own stride interval. Changing the project default will not replace a character's explicit override. Use Character Designer to check that case.

Game Resolution

Viewport Width and **Viewport Height** establish the base game resolution. The example uses **1280 x 720**, a 16:9 shape.

1. Choose the aspect ratio and base size before carefully arranging your GUI.
2. Enter both dimensions and apply.
3. Test a room with the HUD, a dialogue, inventory, and a menu visible.
4. Check another window size before treating the layout as finished.

These are game dimensions, not the size of the editor's buttons or text. Use **UI Size** for the editor interface. A room can also have a viewport-size override, and rendering quality can reduce the 3D

room's internal resolution. Chapter 12 explains how the GUI and overlay can retain the global design size.

4. Choose the starting player and opening sequence

The starting room follows the starting character's location. There is no separate “start scene” selector to keep synchronized with that character.

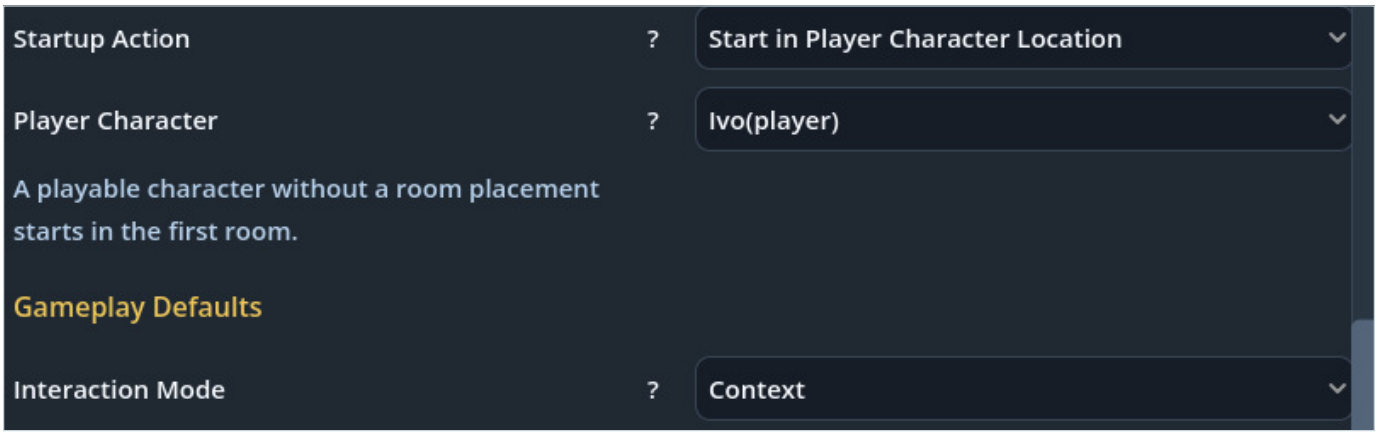

Figure 3. Player Character chooses a playable definition. Startup Action chooses what happens before ordinary play begins.

Start in Player Character Location

1. Create the character in Character Designer.
2. Make it eligible for player control and apply the definition.
3. Place it in the intended starting room.
4. Return to **Project Settings > General > On Game Startup**.
5. Choose it in **Player Character**.
6. Set **Startup Action** to **Start in Player Character Location**.
7. Apply, save, and use Play / Restart.

A playable character without a room placement starts in the first room. Explicit placement is easier to understand and maintain. **Automatic** lets the game's playable/default-character selection resolve the choice; selecting the intended character is clearer when several are playable.

If only one name appears, check the other character's playable eligibility in Character Designer. A character can qualify through its playable/default-player flags or player role. For an ordinary NPC, keep its role and flags consistently non-playable.

Run a cutscene first

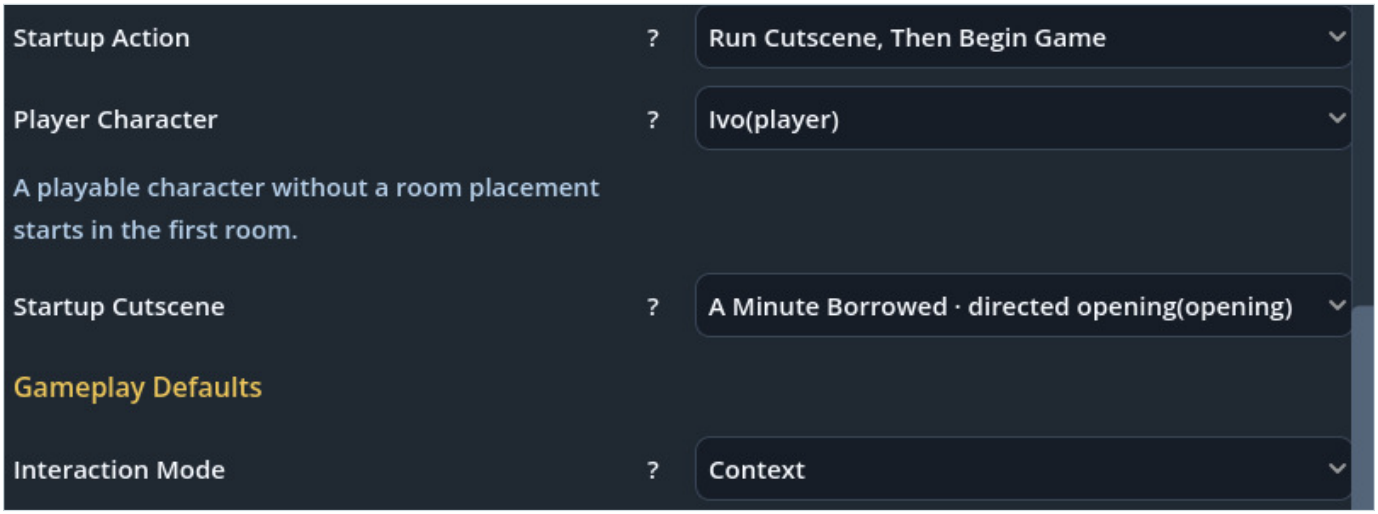

Figure 4. Choosing the cutscene startup mode reveals Startup Cutscene. The selection references an existing cutscene.

Choose **Run Cutscene, Then Begin Game**, then select **Startup Cutscene**. Create and test that cutscene before assigning it here. Selecting its name does not create the cutscene or repair its referenced rooms, actors, and cameras.

After the startup sequence, gameplay begins at the active player's location. A sequence may visit a temporary room; that does not automatically make the temporary room the player's permanent start. Place or move the player deliberately.

Run a script first

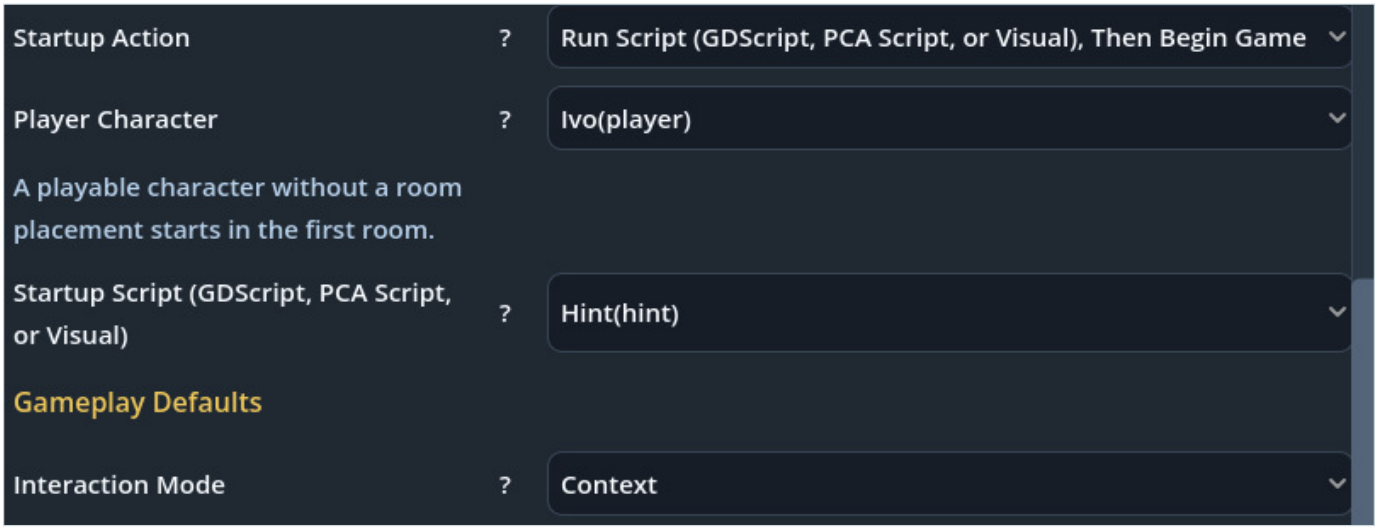

Figure 5. The script startup mode reveals the shared Startup Script selector. The pictured record is an existing example, not a required startup script.

Choose **Run Script (GDScript, PCA Script, or Visual), Then Begin Game** and select **Startup Script**. Use a short, intentional setup routine: initialize a variable, select a playable character, or run an opening action sequence.

A startup script can switch the active character. If it does not, the configured project player remains the intended choice. Avoid scripts that wait forever for an event that can only happen after gameplay has begun.

What runs first?

The ordered startup splashes play before the normal project startup action. An optional authored GUI splash and opening main menu can follow those media cards. Starting a new game then runs the selected startup mode before ordinary gameplay begins for the active player. The game also has its startup/control script hooks. Keep a particular opening sequence in one place so it is not accidentally launched twice by a splash-related script and a startup hook.

To test this, start from Play / Restart and watch the entire opening. Test Scene is useful for the room itself, but it does not establish that the complete startup order is correct.

5. Choose interactions, inventory, and 2D defaults

Interaction Mode

This is the game's starting way of using hotspots and verbs. It is different from **Movement Profile**, which determines how the character travels.

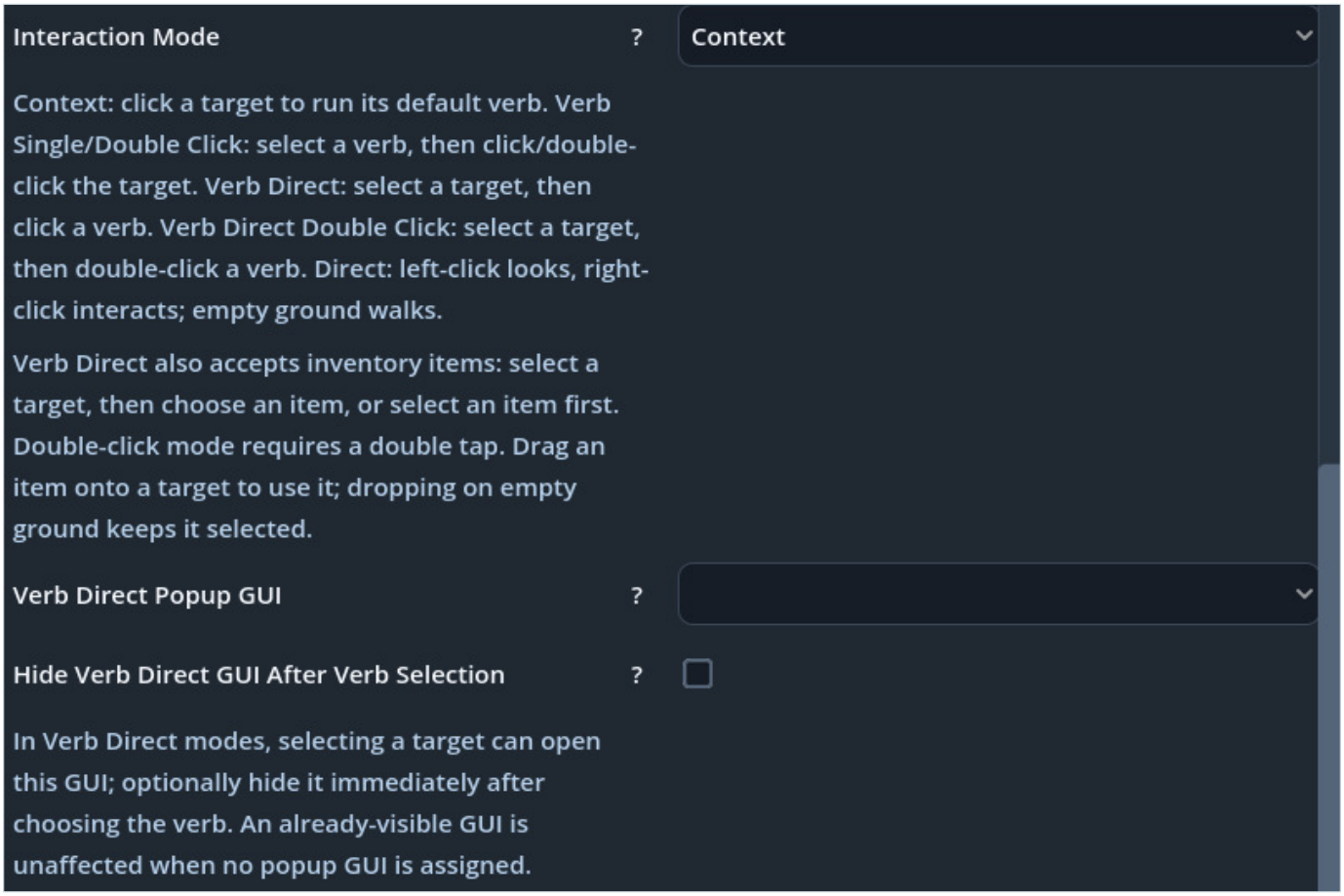

Figure 6. Interaction Mode and the optional Verb Direct popup belong to Project Settings. Movement profiles are configured in Global Settings.

Mode	How the player uses it
Context	Click a target to run its default verb.
Verb Single Click	Choose a verb, then click the target.
Verb Double Click	Choose a verb, then double-click the target.
Verb Direct	Choose a target, then choose the verb.
Verb Direct Double Click	Choose a target, then double-click the verb.
Direct	Left-click looks; right-click interacts; empty ground walks.
Cycling Cursors	Cycle through the available verbs/cursors and apply the current one.
Hotspot Then Verb Coin	Select the hotspot, then choose an interaction from the coin.
Verb Coin Then Hotspot	Choose the interaction from the coin before selecting its target.

For Verb Direct, **Verb Direct Popup GUI** can identify an authored verb-selection interface. **Hide Verb Direct GUI After Verb Selection** controls whether that popup closes when the choice is made. Assigning no popup does not necessarily hide an already-visible verb interface.

Verb Direct also supports inventory items. You can select a target before choosing the item, select the item first, or drag an item onto a target. In its double-click mode, remember the double-tap requirement. Test with an actual inventory item and a hotspot that has a response to it.

Inventory Mode

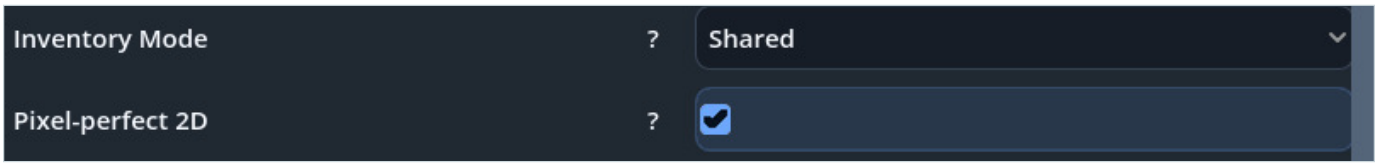

Figure 7. Inventory Mode, Pixel-perfect 2D, and the project-level Autosave flag are farther down General.

Shared gives playable characters a shared inventory arrangement. **Per Character** separates the inventories according to the game's character setup. Choose the mode before designing puzzles that depend on which character carries an item.

Test two playable characters, an initial item, a newly collected item, and a character switch. The presence of two different inventory GUIs alone does not prove the underlying inventories are separate.

Pixel-perfect 2D and Autosave

Enable **Pixel-perfect 2D** for a deliberate pixel-art setup, then inspect the sprites and interface at the target resolution. It is not a substitute for correctly sized sprite frames or sensible scaling.

The **Autosave** flag here is a game-saving default, not a command to save your editor work. In current projects, the more explicit **Global Settings > Player Options > Automatic Game Saving** controls govern the save system and can take precedence. Keep those choices consistent. Chapter 23 covers the author setting, player preference, slots, and thumbnails together.

6. Build the startup splash sequence

Open **Project Settings > Startup Splashes**. This list controls the ordered image/video cards shown before the normal startup action.

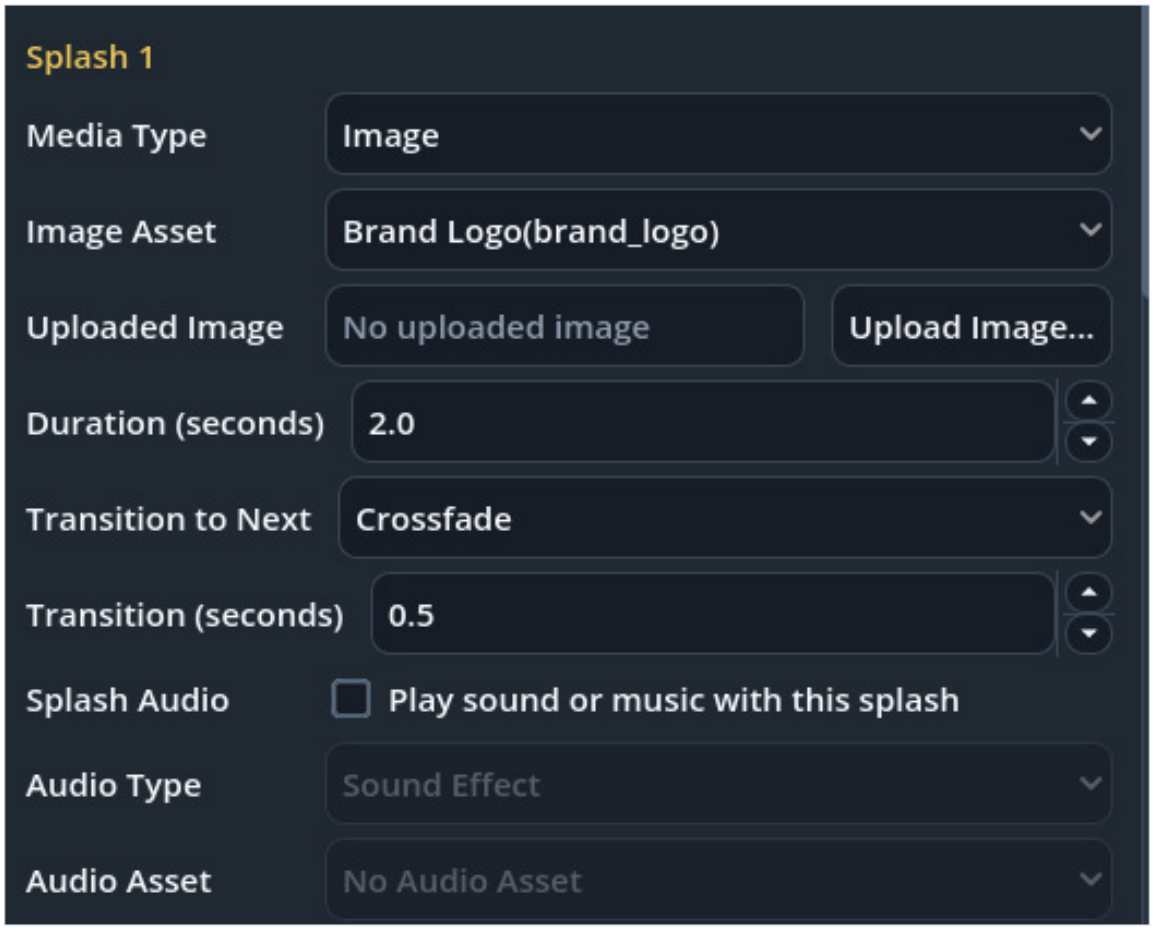

Figure 8. Each splash has its own media choice and timing. The sequence plays from top to bottom.

Add an image card

1. Import the image as an Image asset, or use the row's upload command for standalone media.
2. Click **Add Splash Screen**.
3. Set **Media Type** to **Image**.
4. Choose **Image Asset**, or use **Upload Image...**.
5. Enter **Duration (seconds)**.
6. Choose **Transition to Next** and its duration.
7. Choose **Skip Input**.
8. Apply, save, and start the whole game.

A simple sequence might be a studio logo for two seconds followed by a title image for two seconds. The pictured example uses the bundled Brand Logo and Brand Splash assets. Use your own cleared artwork for your own game.

Duration, transitions, and order

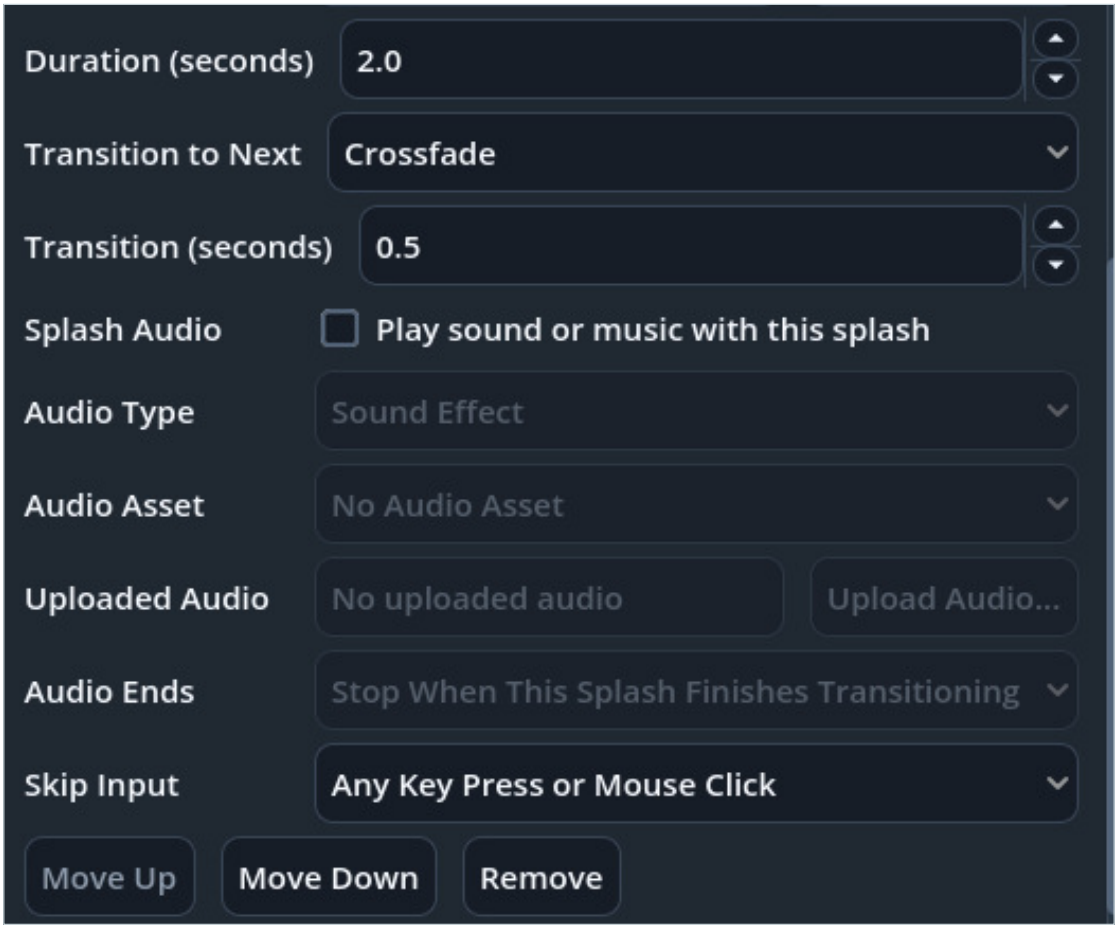

Figure 9. The row holds its own duration, transition, skip input, and Move Up/Move Down/Remove controls.

Cut changes immediately. **Fade Through Black** passes through a black frame. **Crossfade** blends the images. The four wipe directions reveal the next card from the chosen edge. The transition time is separate from the card's authored display duration.

Use **Move Up** and **Move Down** to change the sequence. **Remove** removes that entry from the list. Check the opening after reordering, especially when audio continues across the transition.

Skip Input can allow neither input, keyboard, mouse, or both. Choose intentionally and test it. A long unskippable studio card gets tiresome rather quickly once the player has seen it several times.

Video cards

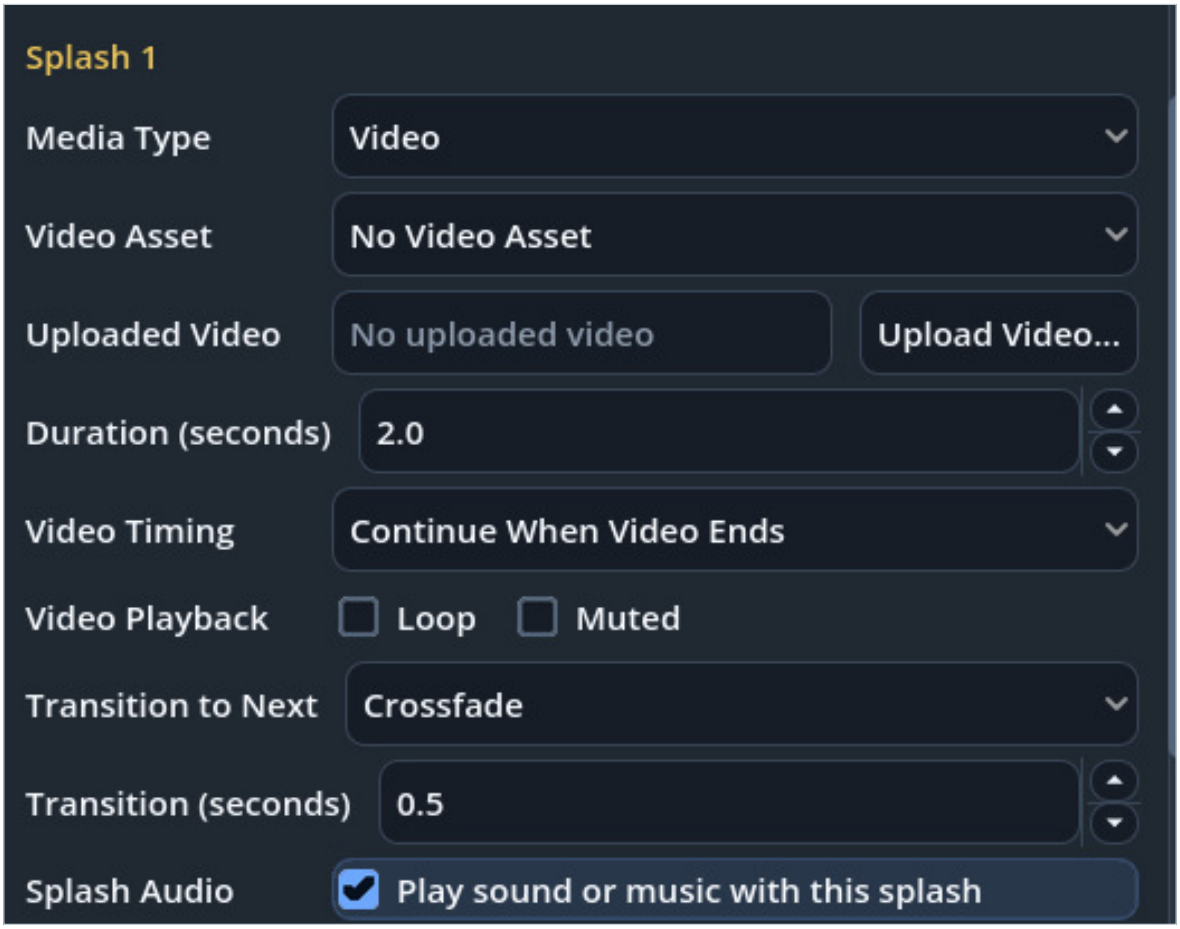

Figure 10. Video adds end-of-video/fixed-duration behavior, looping, and muting controls. This reference row has no video asset assigned.

Set Media Type to **Video**, choose/import the video, and decide between **Continue When Video Ends** and **Use Authored Duration**. Check the loop and mute choices. The imported video's actual playback support still matters on the target build.

A video card is not a cutscene containing the player's room actors. Use a startup cutscene for those authored in-engine actions. For a prerecorded logo/movie, use the video card and test its timing.

Optional splash sound or music

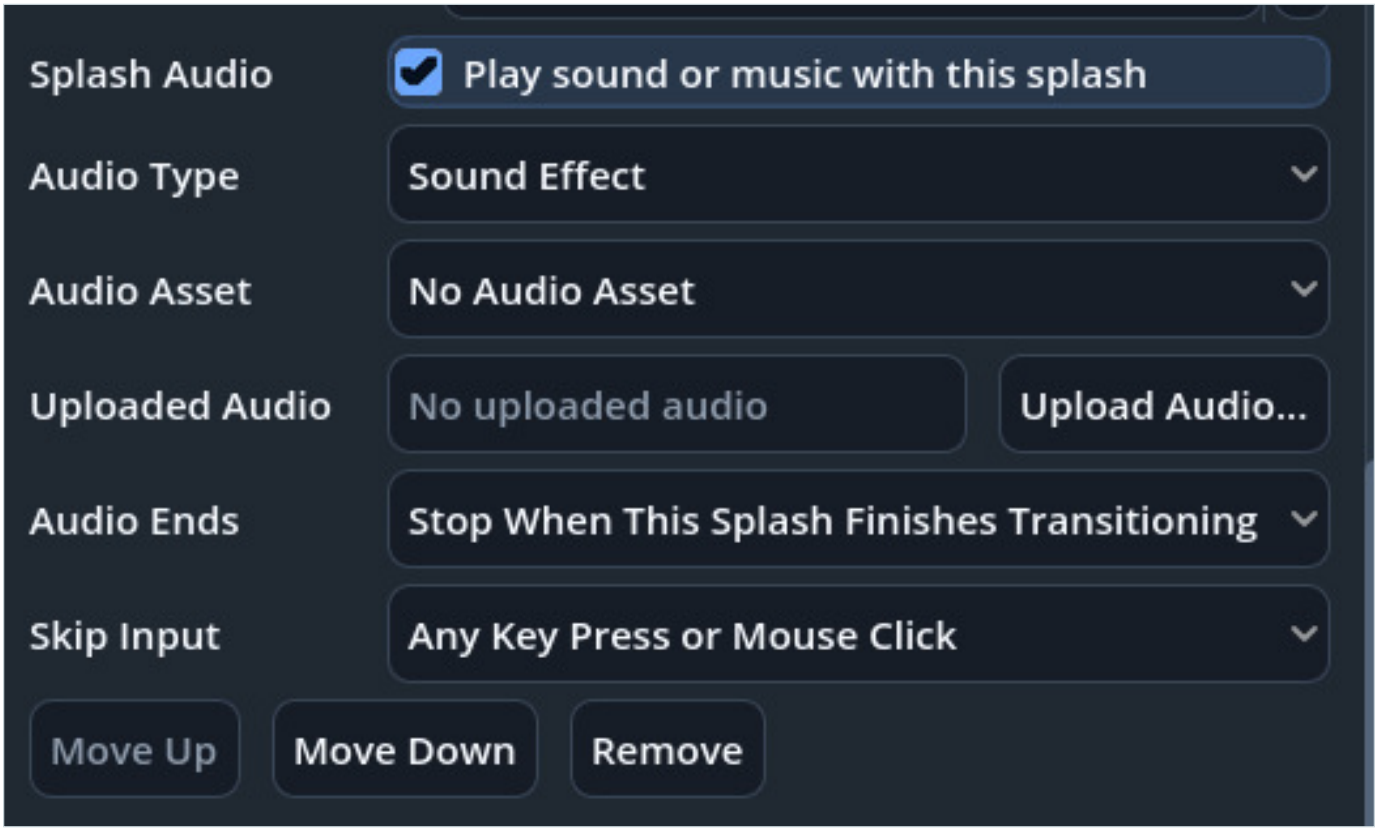

Figure 11. Enable Splash Audio to make its audio selectors active. Audio Ends determines whether playback stops with this card or continues.

Enable **Play sound or music with this splash**. Choose Audio Type and an Audio Asset, or upload the audio. Decide what **Audio Ends** should do:

Choice	Meaning
Stop When This Splash Finishes Transitioning	Ends the accompaniment with that card's transition.
Keep Playing Until the Audio File Ends	Lets the recording continue naturally.
Keep Playing Until a Script Stops It	Requires a deliberate script/audio stop later.

A video may already contain audio. Do not add a second copy of the same soundtrack unintentionally. Test skipping as well as waiting for the card: continuing audio needs a deliberate end in both paths.

7. Name the distribution files and choose the icon

Open **Project Settings > Packaging**. These fields describe your game's build names and icon. They do not package PCA 3D Adventure Studio itself.

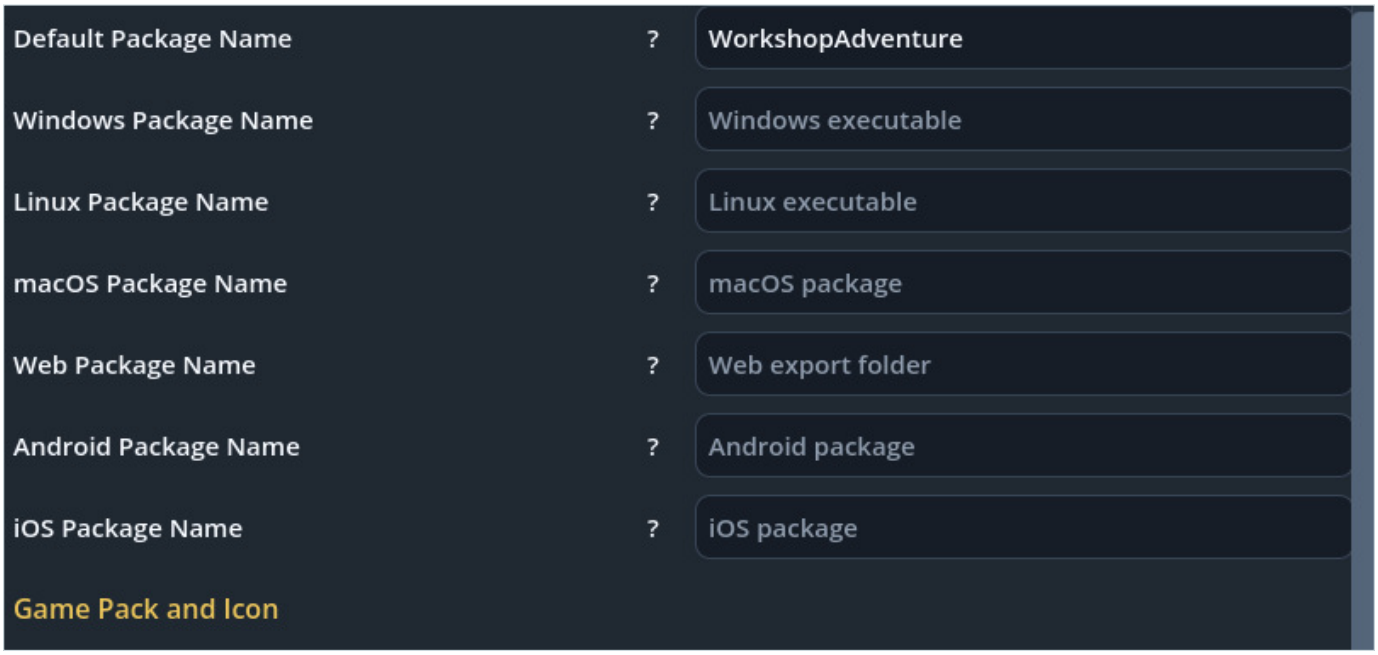

Figure 12. A platform name overrides Default Package Name. Blank Default falls back to the project's title.

Package-name fallback

Use a short filename stem, such as `workshop_adventure`. A platform-specific name takes priority for that platform. If it is blank, the builder uses **Default Package Name**. If that is blank too, it derives the name from the project title.

Windows, Linux, macOS, Web, Android, and iOS have separate fields. The builder sanitizes names and adds the appropriate extensions. Do not depend on punctuation or file extensions typed into a stem field being preserved exactly.

The readable Title and the filename can be different. **The Small Workshop** can be the game title while `workshop_adventure` is the distribution name.

Game PCK Filename and Game Icon Image

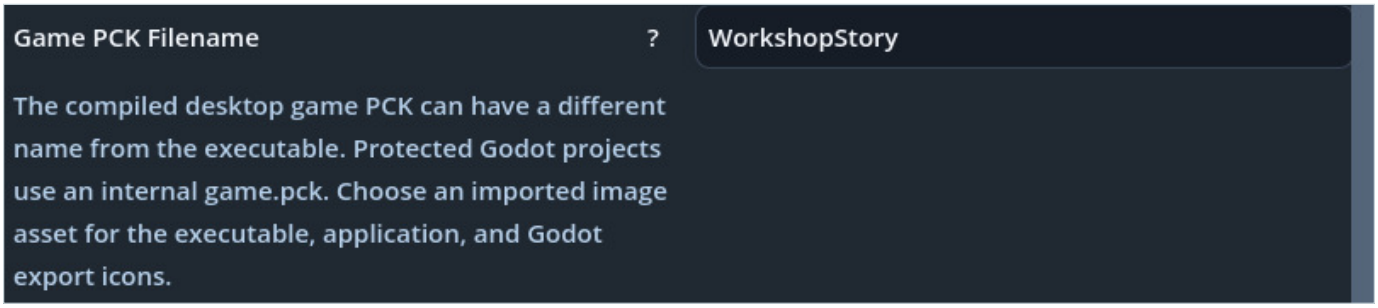

Figure 13. The game-data PCK has its own name. Game Icon Image references an imported Image asset.

Game PCK Filename names the game-data pack. It can differ from the executable and its matching runner pack. Leaving it empty uses the normal default. Keep the generated executable/application, data packs, launch configuration, and supporting files together.

Game Icon Image selects an imported image. Use a clean, sufficiently large image that still reads at small icon sizes. The builder prepares platform icon files from it; the source asset is not changed. Leaving it unassigned keeps the default icon behavior.

An icon and a package name do not replace platform build requirements. Build Game and Build Tools handle the actual target/export toolchain. Mobile signing and SDK configuration, for example, are not supplied by filling in an Android package-name field. Build-tool locations are device preferences, so another machine may need its own setup.

After a build, inspect the output names and icon, then launch the complete output from a separate folder. Renaming or distributing only the executable can break its relationship with its packs and launch data.

8. Add game languages and hand translations to other people

Use **Game Languages & Translations...** above Project Settings. This changes game text, translated assets, and voice mappings. The editor toolbar's **Language** changes the editor interface; it is a separate choice.

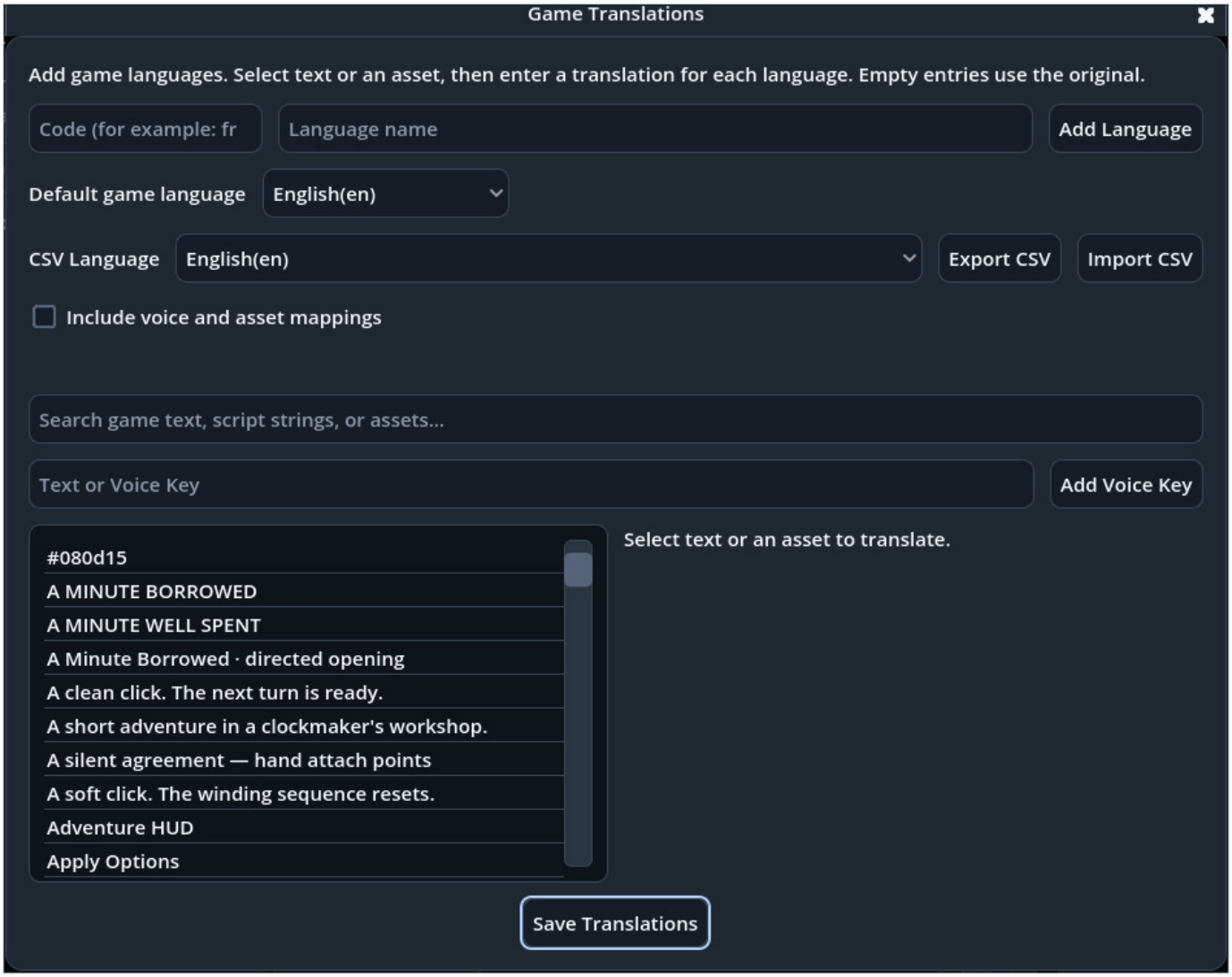

Figure 14. The game-localization window lists languages and translatable entries. CSV Language selects the language used for export/import.

Add and test one language

1. Enter the language code, such as `fr`.
2. Enter its readable name, such as French.
3. Click **Add Language**.
4. Select a text entry and enter the translation in that language's field.
5. Choose the intended **Default game language**.
6. Click **Save Translations**, then save the project.
7. Start a fresh preview and check the actual line.

Empty entries use the original. Use the search field to find text, script strings, or asset references. An asset translation points to the appropriate translated asset; it does not translate the pixels inside a picture automatically.

CSV collaboration

For people translating in a spreadsheet:

1. Add the target language in the project first.
2. Choose it in **CSV Language**.
3. Leave **Include voice and asset mappings** off for a straightforward text-only handoff.
4. Click **Export CSV** and give that language's file to the translator.
5. Have the translator edit the **translation** column while preserving the identifiers and other columns.
6. Select the same CSV language and use **Import CSV** when the file returns.
7. Review the change/conflict preview before accepting it.
8. Save Translations, then save the project.

The exported columns include `format`, `language`, `type`, `key`, `source`, `translation`, and `original`. The last column retains the original exported translation for conflict detection; it is not another place to put a new translation. Preserve the header, keys, language, and format marker.

Give separate files to different languages. If several people share one language, agree which entries each person owns and review conflicts when combining their work. Do not casually enable **Overwrite conflicting translations** when someone else has already revised those entries in the project.

Voice and asset mappings are useful for a coordinated localization pass, but their cells refer to actual assets/recordings. A translated voice needs its own recording and lip-sync timing, as explained in the Character Designer guide.

9. Choose a renderer and quality level

Open **Global Settings > General > Rendering**. Start here before spending time adjusting effects that your active renderer may not support.

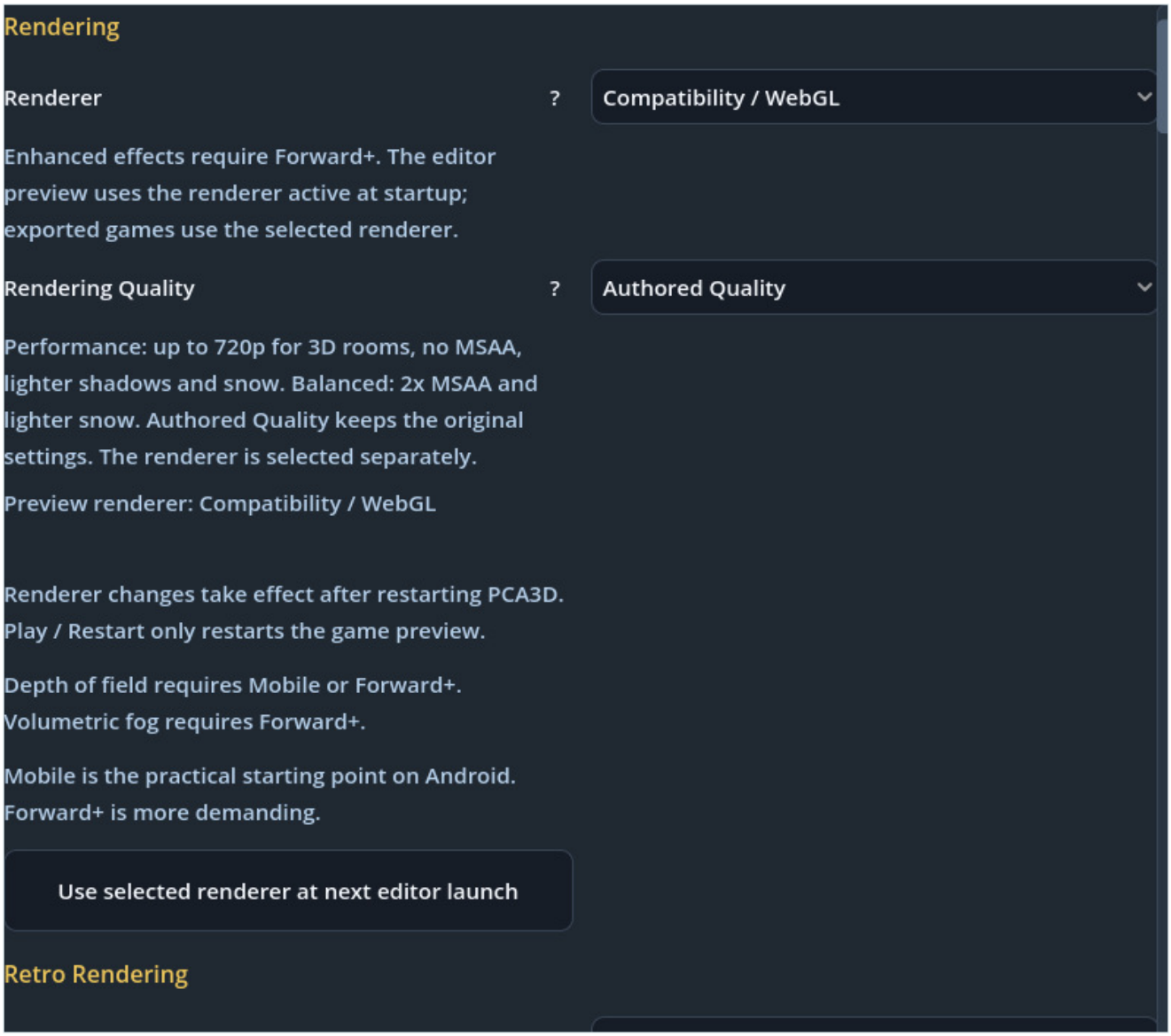

Figure 15. Renderer and Rendering Quality are separate choices. The page also reports the renderer actually running the preview.

The four renderer selections

Selection	Practical use
Compatibility / WebGL	A useful starting point for older/integrated graphics and browser-oriented work. Some advanced effects are unavailable.
Mobile	A lighter modern rendering path. It also supports native depth of field where the presentation allows it.
Forward+	The modern desktop path for advanced lighting/effects on suitable hardware.
Forward+ Enhanced	Forward+ with PCA's optional enhanced lighting/reflection/occlusion configuration. It still needs an active Forward+ backend.

These are not promises of identical appearance or speed on every device. Build and test on the intended hardware. If a project crashes or performs poorly only after a demanding room loads, begin with a simpler renderer/quality combination and add effects one at a time.

The screenshots in this book use **Compatibility / WebGL**. Enhanced controls are shown as configuration examples; their presence in a form does not mean their effects are active on that backend.

Apply the game choice and restart the editor correctly

1. Choose Renderer and apply the form.
2. Save the project. This records the selected renderer for game/build output.
3. If you want the editor preview to use that backend too, click **Use selected renderer at next editor launch**.
4. Save, close PCA, and launch it again.
5. Revisit the page and check **Preview renderer**.

Play / Restart restarts the game preview inside the existing application. It cannot replace the application's active rendering backend. Forward+ and Forward+ Enhanced use the same underlying Forward+ backend; the Enhanced choice adds the project's optional configuration.

Native depth of field requires Mobile or Forward+. Native volumetric fog requires Forward+. Ordinary depth fog and lighting are separate from volumetric fog. For a layered/transparent presentation, check the scene/camera's own supported-effect information as well.

Rendering Quality

Quality	What changes
Authored Quality	Keeps the intended authored rendering settings.
Balanced	Uses lighter shadow/snow handling and 2x MSAA when antialiasing is enabled.
Performance	Caps a 3D room's internal render size at up to 1280 x 720, disables MSAA, and lightens shadows/snow.

Quality and renderer are independent. Performance does not choose Compatibility for you. It also does not delete models, animations, actions, or other project content. The internal 3D-room cap is not the same thing as changing the project's GUI design resolution.

For a first performance pass, compare the same room, camera, and activity under each quality level. Stop and restart the preview between changes so you know which configuration you are judging.

10. Use cinematic grading and recognize the two retro paths

General contains both **Retro Rendering** and **Cinematic Presentation**. Keep their purposes clear before combining them with the scene-shader stack.

Retro Rendering



Figure 16. Retro Rendering has style, strength, pixel/jitter, and fog controls. It is separate from the ordered screen-effect stack.

This material-oriented retro setup offers Disabled, PSX, N64, and Horror Fog choices. Its controls include Strength, Retro Pixel Size, Vertex Jitter, and retro fog amount/start/end/color. Objects with custom shaders can retain their own appearance.

For the **PSX, N64, and horror-fog screen effects**, go to **Default Global Scene Shader Stack** farther down General and choose them in a Scene Layer. Chapter 19 gives that workflow and its dedicated parameters. Avoid enabling both retro routes while you are trying to learn which one creates a particular change.

Cinematic Presentation

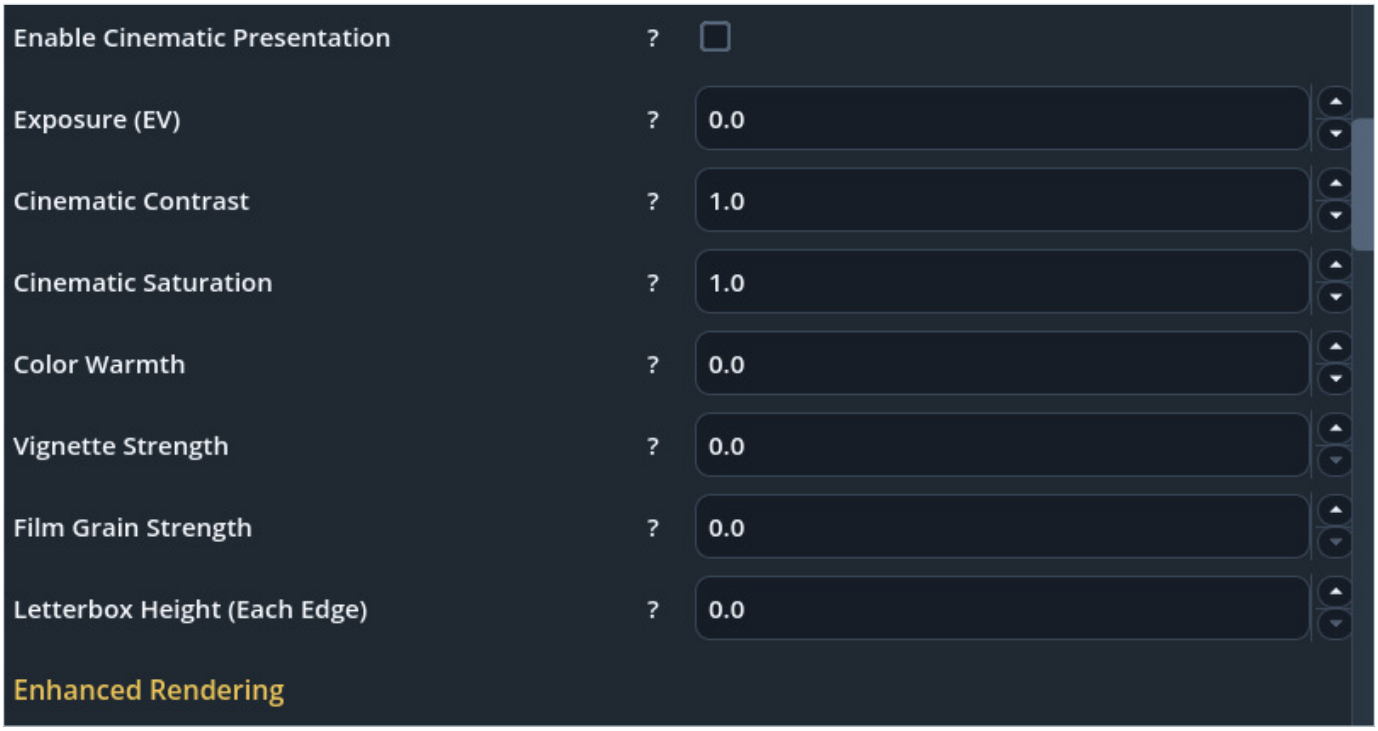

Figure 17. Cinematic grading adjusts the presented world image after lighting. The enable checkbox controls whether this group is used.

Control	Use it to
Enable Cinematic Presentation	Turn the grading treatment on or off.
Exposure (EV)	Brighten or darken the presented image.
Cinematic Contrast	Change the difference between dark and light areas.
Cinematic Saturation	Reduce or increase color intensity.
Color Warmth	Push the treatment warmer or cooler.
Vignette Strength	Darken the edges.
Film Grain Strength	Add visible image grain.
Letterbox Height (Each Edge)	Cover the world's upper/lower edges for a widescreen treatment.

Start gently: enable the treatment, leave most values neutral, and adjust one control. Inspect the darkest useful object, readable faces, and a bright light source. A dramatic screenshot is not much use if the player can no longer see the key on the table.

Letterboxing belongs to the world presentation rather than automatically covering the GUI. Rooms can inherit or override cinematic settings. If one room ignores the global change, inspect its override mode before increasing every value.

11. Configure lighting, reflections, shadows, and antialiasing

Enhanced Rendering

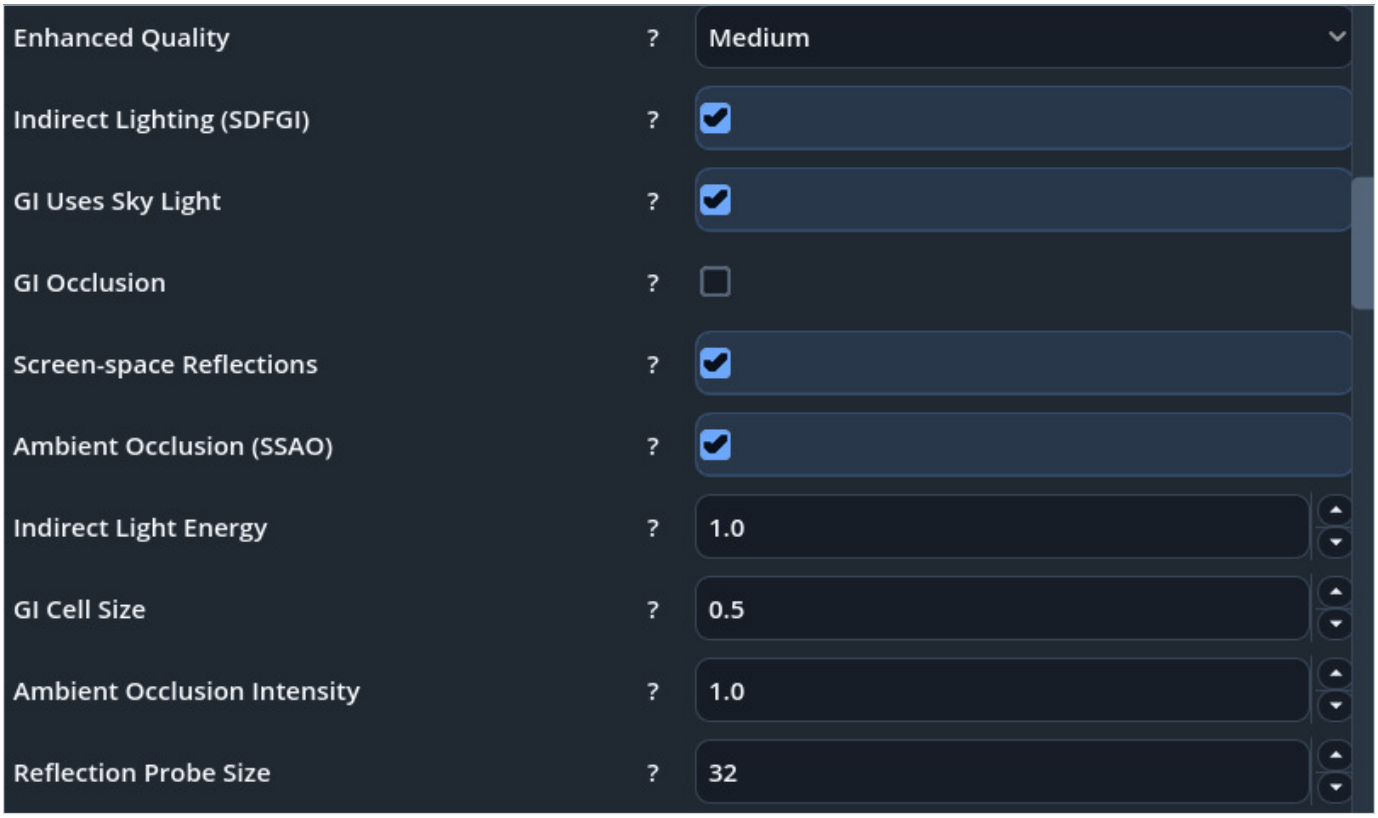

Figure 18. Enhanced Rendering exposes the optional Forward+ Enhanced configuration. These effects need the appropriate selected profile and active backend.

Field	What to consider
Enhanced Quality	Overall cost/detail tradeoff for the enhanced configuration.
Indirect Lighting (SDFGI)	Adds indirect lighting based on static scene geometry.
GI Uses Sky Light	Includes sky illumination in that indirect-light setup.
GI Occlusion	Adds occlusion to the GI solution where supported.
Screen-space Reflections	Uses information visible to the screen; it has screen-space limits.
Ambient Occlusion (SSAO)	Adds contact/depth shading rather than a new light.
Indirect Light Energy	Scales the GI contribution.

Field	What to consider
GI Cell Size	Controls the spatial detail/cost of the GI representation. Smaller is more demanding.
Ambient Occlusion Intensity	Scales the darkening effect.
Reflection Probe Size	Sets the shared size reference for room-probe capture.

Use static floors/walls for the first GI test. Moving objects can receive indirect light without continuously rebuilding the GI shadows of the room around them.

A reflection probe is not a live mirror

Room Reflection Probe captures nearby room geometry when the room loads. Reflective materials also need suitable roughness/metallic/reflection settings and something useful to reflect. Lower roughness makes a suitable reflection sharper.

That capture is not a guarantee of a live, accurate reflection of a walking character. Screen-space reflections have their own visibility limits. For a deliberate mirror surface, use the scene/object mirror workflow described in the Scene Designer guide and test it from the player's actual camera.

Move the room's reflection probe in the room Inspector when its capture position needs correction. Making every object metallic is not a substitute for a useful capture or a mirror setup.

The regular rendering controls



Figure 19. Shadows, Antialiasing, Tone Mapping, Ambient Color, and Ambient Intensity sit below the enhanced group.

Shadows controls the shared shadow choice; individual lights and objects still have their own relevant settings. **Antialiasing** smooths edges, with the selected quality level affecting the MSAA workload.

Tone Mapping offers Linear, Reinhard, Filmic, and ACES. Compare the actual result with your intended renderer and scene lighting. It is a presentation choice, not a cure for a missing material texture.

Ambient Color and **Ambient Intensity** affect the shared ambient-light treatment where used. Start with a restrained intensity and inspect objects away from direct lights. Very strong ambient light can flatten the room and hide the shadow/shape differences you were trying to create.

12. Keep the GUI size steady and show hover names

Viewport Presentation

A room can choose a viewport size different from the project's base resolution. These two checkboxes decide whether the GUI and screen overlay retain the global design space when the world viewport changes.

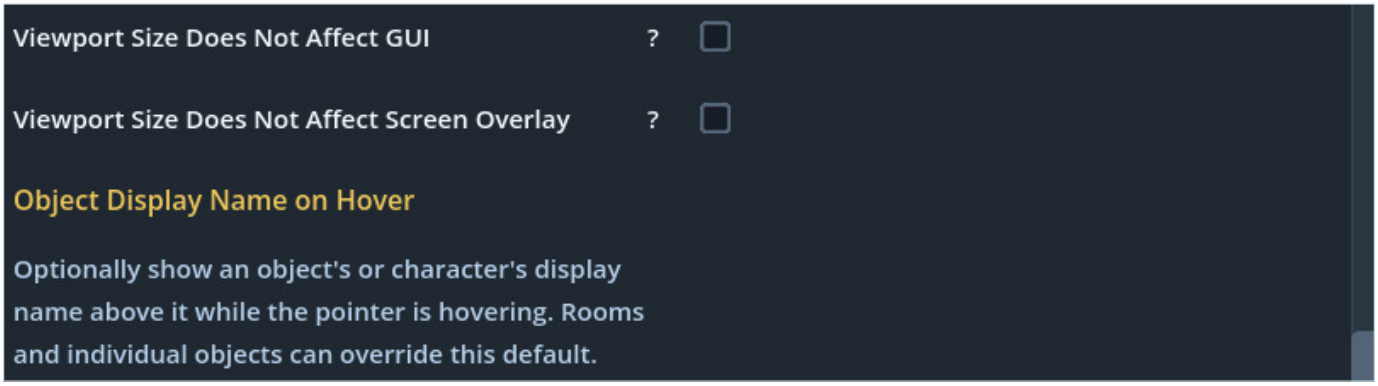

Figure 20. GUI and overlay size independence are separate choices. They do not resize the room's navigation or floor.

Enable **Viewport Size Does Not Affect GUI** when a special room should change the world presentation while keeping the interface on the global design size. Use **Viewport Size Does Not Affect Screen Overlay** independently for the overlay.

For a practical test, compare an ordinary room and the overridden room with the same menu open. Check button positions, portrait framing, overlay edges, and click/touch alignment. Do not judge only the background picture.

These options do not enlarge a 2.5D navigation area or an object's collider. Those belong to the scene and object settings.

Object Display Name on Hover

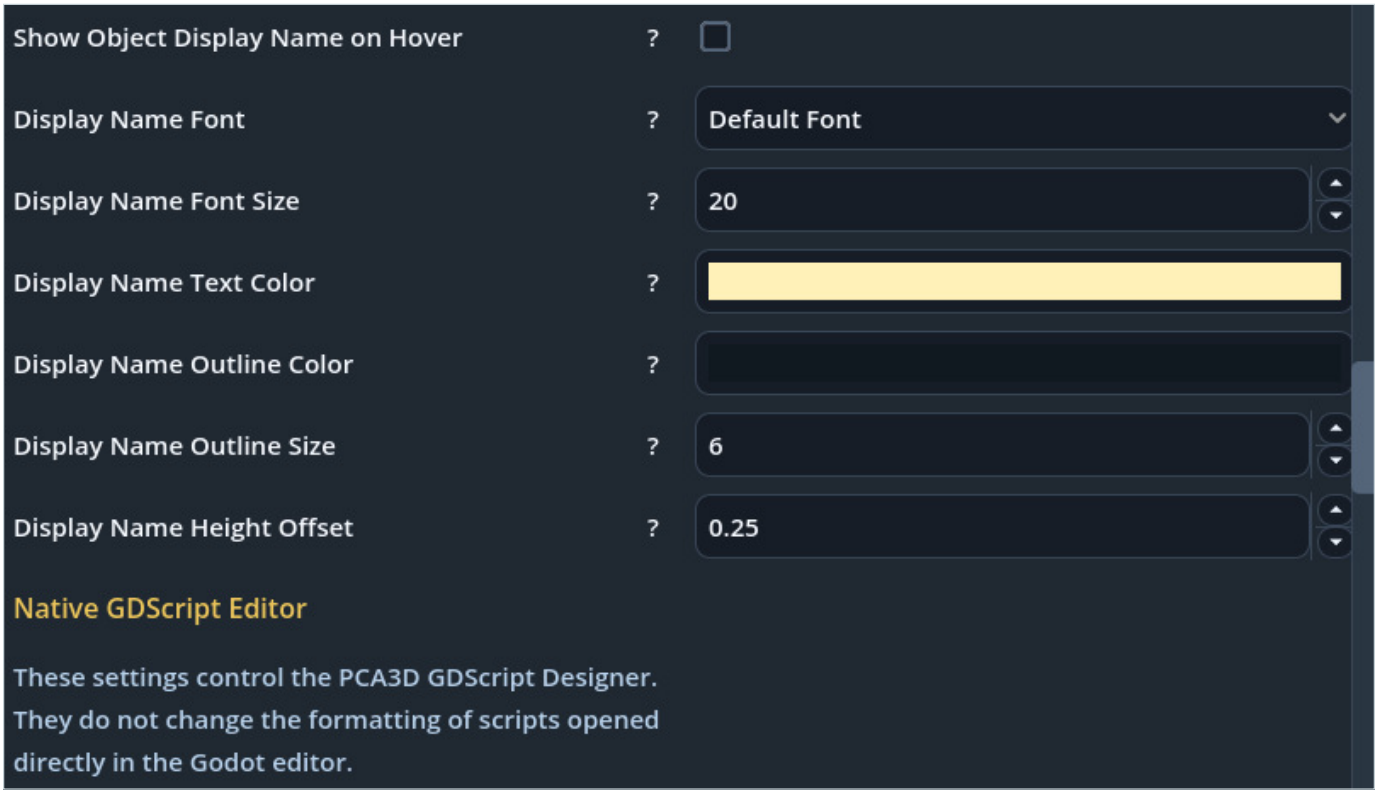

Figure 21. The global hover-name defaults include font, colors, outline, and height. Rooms and individual objects can override them.

Enable **Show Object Display Name on Hover** to show an object's or character's readable name above it when the pointer is over it. Choose a Display Name Font or the default, Font Size, Text Color, Outline Color, Outline Size, and Height Offset.

Start with a readable light text color and enough outline to survive both a bright floor and a dark doorway. Use a small height offset, then check a tall object and a short object. A room or object with its own override may behave differently from this shared default.

Hover labels are not a substitute for a touch/controller interaction cue. On those devices, also configure the proximity/highlight and controller-hint behavior in Adventure & Cursor.

13. Set walking speed and automatic approach

Open **Global Settings > Adventure & Cursor**. **Default Walk Speed** supplies the shared speed for characters using the project default.

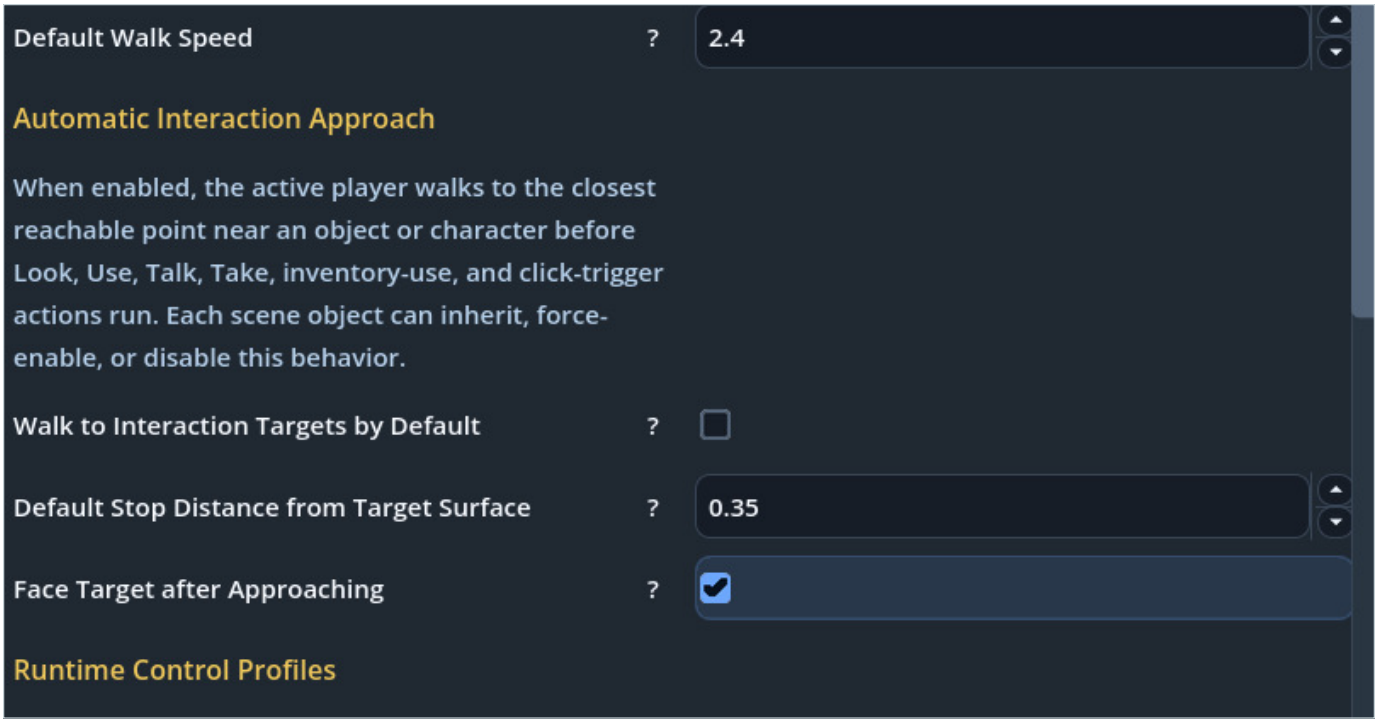

Figure 22. The shared walk speed and automatic approach controls establish sensible starting behavior for interactions.

Test speed in world scale. The example starts around 2.4 world units per second. A character can have its own speed in Character Designer, and a room may have relevant overrides. If changing this value seems to do nothing for one person, check those more specific settings.

Walk to a target before using it

Enable **Walk to Interaction Targets by Default** if Look, Use, Talk, Take, inventory-use, and click-trigger responses should normally approach their target first.

1. Enable the default approach.
2. Set **Default Stop Distance from Target Surface**.
3. Decide whether to **Face Target after Approaching**.
4. Apply and test an object on clear ground.
5. Test a doorway, a large prop, and another character.

The player tries to reach a suitable reachable point near the target. This is not a command to walk through its physical body. An individual object can inherit, force-enable, or disable approach.

If the player stops too far from a door, check the door's collider and blocking settings as well as stop distance. Shrinking an event volume does not necessarily shrink the physical obstruction. The Scene Designer guide covers those dimensions separately.

14. Choose movement profiles and input switching

Movement profiles describe how travel works. Interaction modes describe how a verb/hotspot is chosen. You can change one without rebuilding the other.

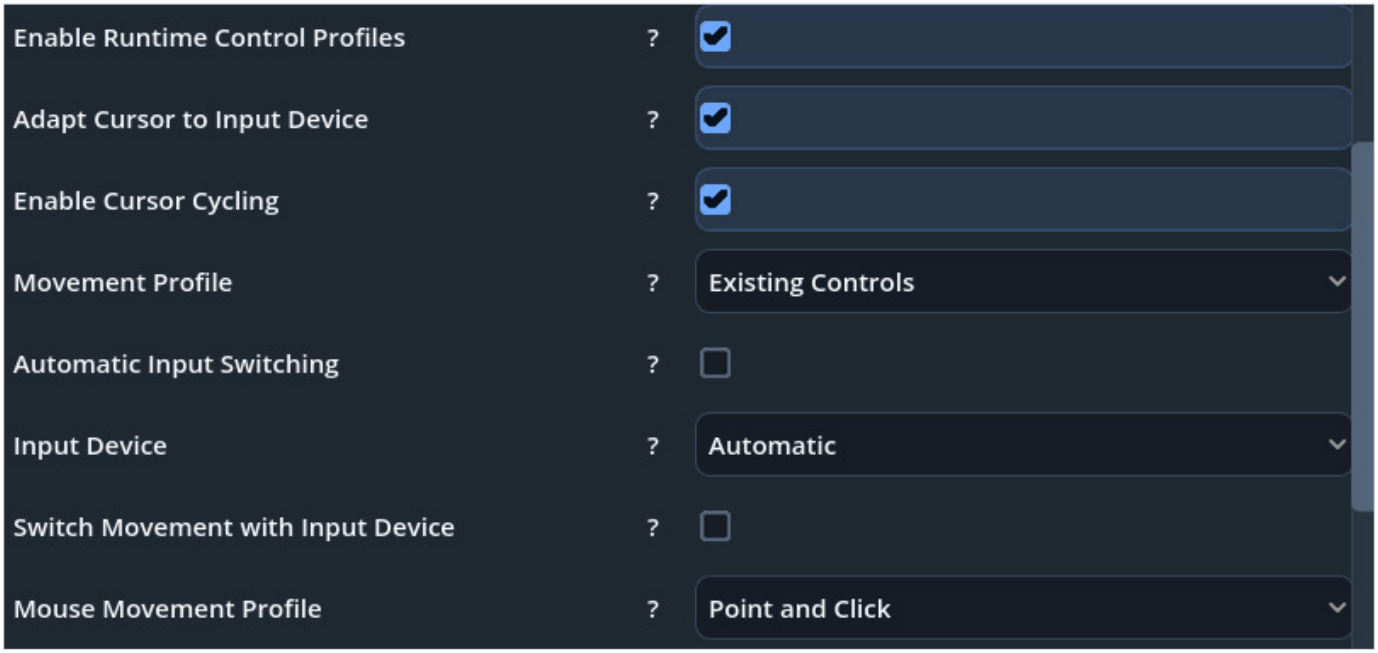

Figure 23. Enable Runtime Control Profiles controls the optional profile system. Existing Controls preserves the established movement path.

The main choices

Movement Profile	What it is for
Existing Controls	Keep the established/legacy control arrangement.
Point and Click	Point at a destination for navigation-based walking.
Direct Movement	Move through direct input such as keys or a stick.
First Person	A first-person movement/look arrangement.

Movement Profile	What it is for
Drag to Move	A drag gesture used for movement.

Start with one profile and one input device. Leave automatic movement switching off until that basic arrangement works. **Enable Runtime Control Profiles** can be turned off when you want the existing control system without these adaptations.

Automatic input and movement switching

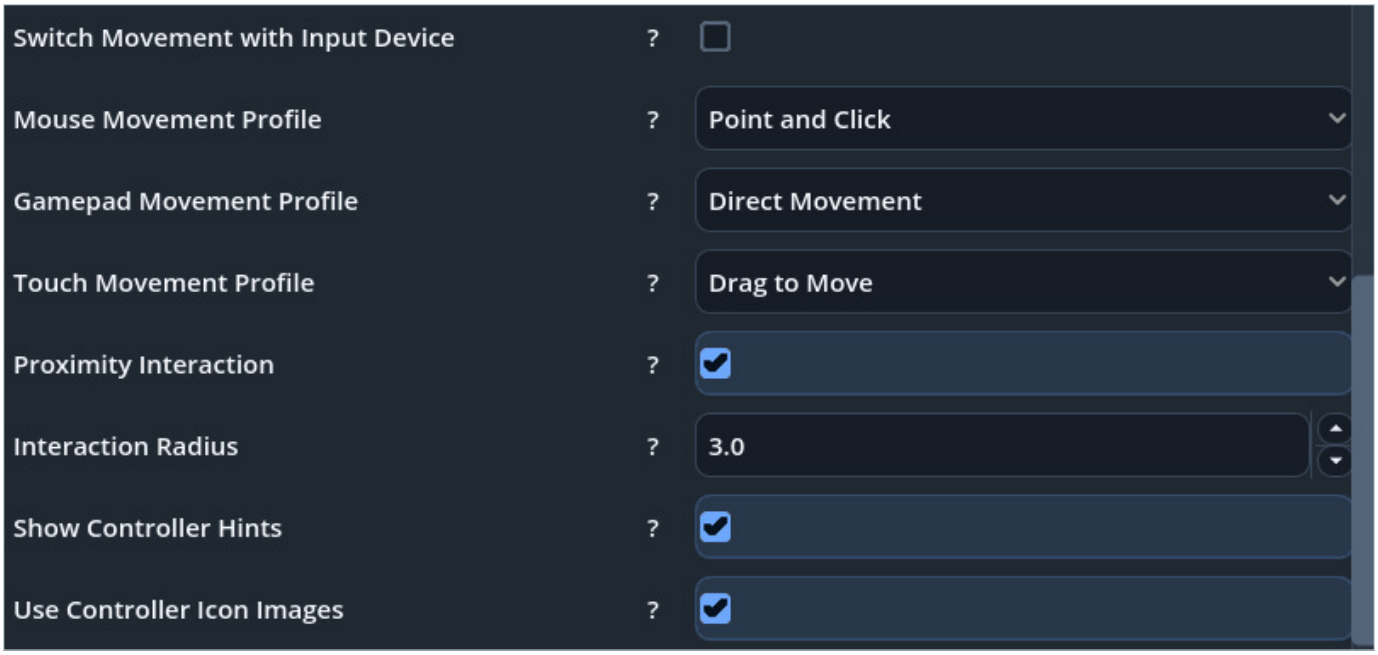

Figure 24. Automatic Input Switching and Switch Movement with Input Device are separate switches. The three per-device profiles are their movement choices.

Input Device can be Automatic, Mouse and Keyboard, Gamepad, or Touch. **Automatic Input Switching** lets detected input change the active device treatment. **Switch Movement with Input Device** additionally selects the authored mouse/gamepad/touch movement profile.

For example, choose Point and Click for mouse, Direct Movement for gamepad, and Drag to Move for touch. Enable the relevant automatic switches, apply, and test a transition between actual devices. **Adapt Cursor to Input Device** controls the associated cursor treatment.

Selecting a gamepad profile does not itself enable the Controllers page's hardware input. Likewise, profile switching does not create navigation in a room that has no walkable area.

Proximity, hints, cycling, and the coin



Figure 25. Proximity and hint controls support non-mouse play. Tune the radius against the scale of the room.

Proximity Interaction and **Interaction Radius** determine the nearby-target assistance. A very large radius can make the character select something across the room. Check two neighboring objects so you can tell which one is active.

Show Controller Hints and **Use Controller Icon Images** control the hints/artwork path. **Enable Cursor Cycling** and **Enable Interaction Coin** allow those mechanisms. The selected interaction mode still determines how they are used.

Drag Threshold helps distinguish an intentional drag from a small pointer/touch movement. **Gamepad Look Speed** sets the look response for relevant control modes. Test short movements before increasing sensitivity.

15. Configure keyboard movement and cursors

Keyboard Player Movement

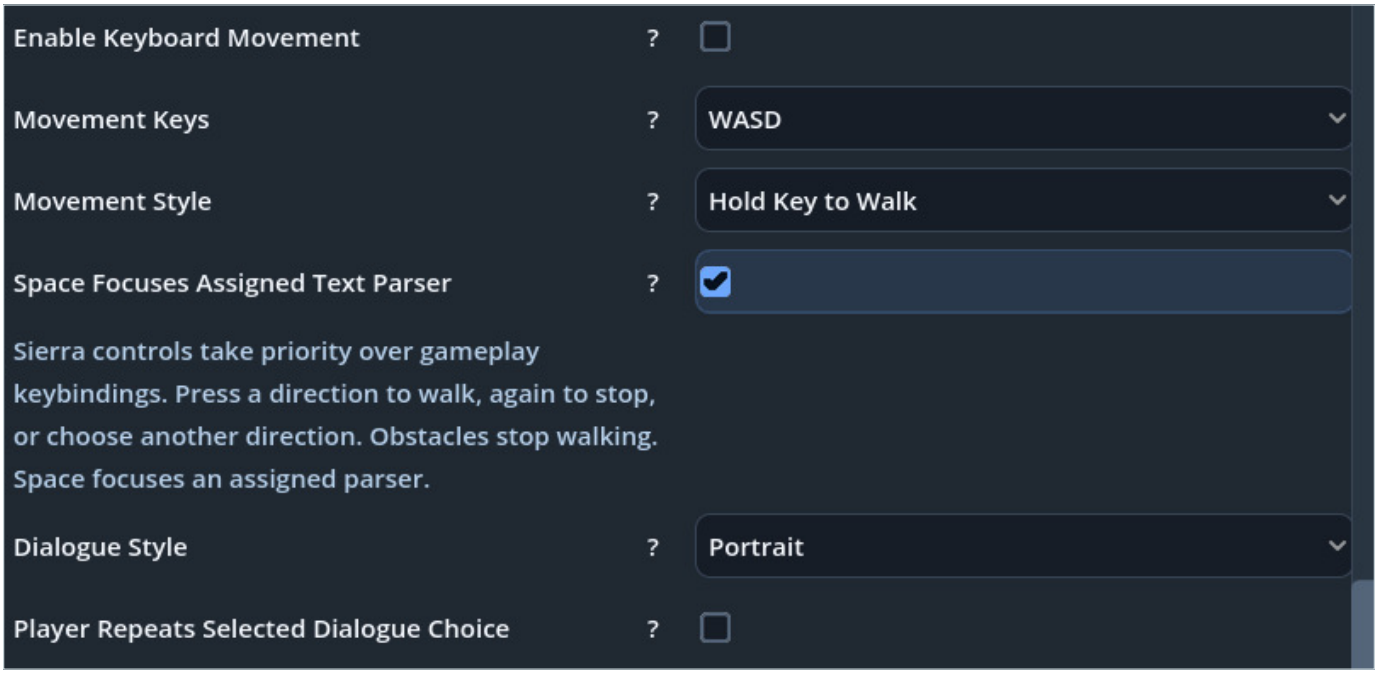

Figure 26. Keyboard movement can be added to navigation-based play. Movement Style chooses held-key or Sierra-style behavior.

Enable **Enable Keyboard Movement**, then choose WASD, Arrow Keys, or both. **Hold Key to Walk** moves while a direction is held. **Sierra AGI: Press to Walk / Stop** starts movement on a direction press; another press can stop it or choose a new direction. Obstacles can stop that movement.

Space Focuses Assigned Text Parser gives Space that role when the project has an assigned parser. Sierra controls take priority over gameplay keybindings, so check for conflicts before assuming a different shortcut is broken.

This option still uses the room's navigation. It does not make an unreachable floor reachable. For ordinary gameplay shortcuts, visit the separate **Keybindings** designer; that is not the same thing as choosing this movement-key set.

Cursor options

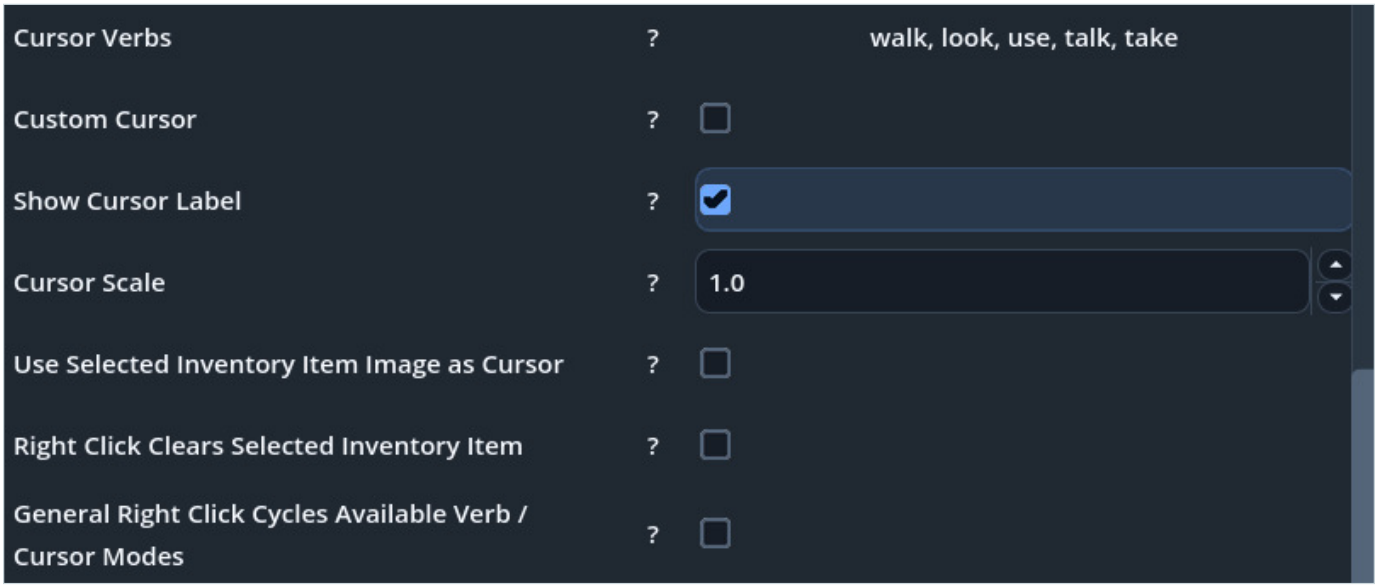

Figure 27. Cursor Verbs selects eligible verbs. The remaining controls choose the cursor's appearance and inventory/right-click behavior.

Control	What to check
Cursor Verbs	Select the actual authored verbs that should be available.
Custom Cursor	Use the game's cursor presentation. Create/configure cursor records first.
Show Cursor Label	Display the cursor's label where supported.
Cursor Scale	Size the cursor sensibly at the intended resolution.
Use Selected Inventory Item Image as Cursor	Show the selected item's picture as the pointer.
Right Click Clears Selected Inventory Item	Let right-click cancel the selected item.
General Right Click Cycles Available Verb / Cursor Modes	Cycle when that right-click has not been handled by something more specific.

Object-specific right-click responses take priority. A selected inventory item can also consume a right-click through the cancellation behavior. Do not assume the general cycling checkbox will override those actions.

Test an empty floor, a hotspot, and a selected inventory item. That small test catches most “right-click behaves differently here” misunderstandings.

16. Choose the dialogue presentation and response rules

These controls are also on **Adventure & Cursor**.

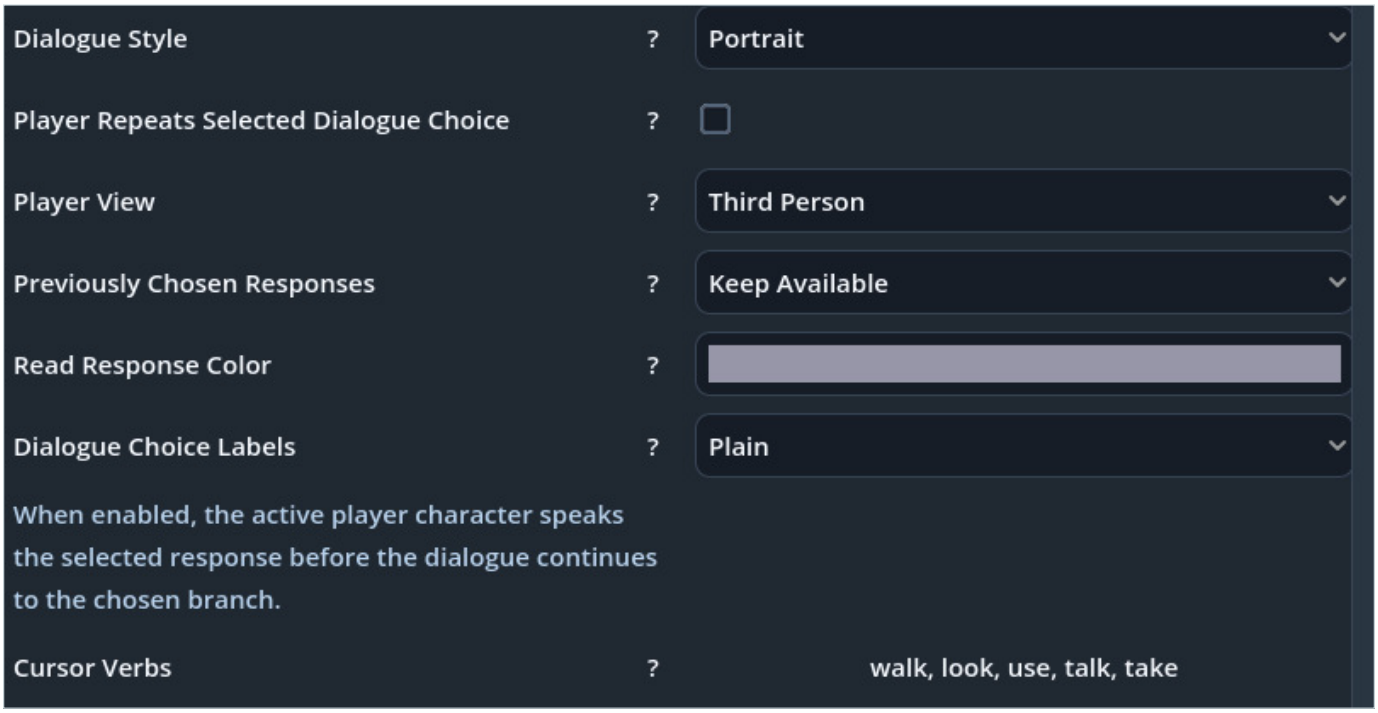

Figure 28. Dialogue Style, repeated player responses, read-response rules, and choice labels are shared dialogue defaults.

Dialogue Style offers Portrait, Classic, Speech Bubble, Subtitle, and Cinematic presentations. A custom system GUI may provide its own layout, so a style change is not guaranteed to redesign that authored GUI.

Player Repeats Selected Dialogue Choice makes the active player speak the selected answer before the conversation follows that branch. Check that the repeated line has the desired voice/portrait arrangement; otherwise the extra step can feel like unnecessary duplication.

Remembering choices

Previously Chosen Responses can keep a response available, change its color, disable it, or both recolor and disable it. **Read Response Color** supplies the alternate color. Use a readable difference that works against the actual GUI background.

Dialogue Choice Labels chooses Plain, Bulleted, or Numbered labels. Individual dialogue records can supply relevant authored behavior, so test the actual conversation rather than only a generic message.

Disabling a previously selected response can change a puzzle's available routes. Make sure the player still has a way to leave the conversation and obtain necessary information. Use dialogue-state actions when a later puzzle step should deliberately add, remove, or reset a choice.

Player View

Player View chooses Third Person or First Person / FPS Style. It appears near the dialogue controls, but it concerns the player's view, not the shape of a speech panel.

Keep this consistent with the intended movement profile and camera setup. A first-person profile, a first-person view, and mobile first-person sticks are related settings in different groups. Configure and test them as one intended control arrangement.

17. Set up the native GDScript editor

Scroll down **Global Settings > General** to **Native GDScript Editor**. These choices are stored in the project but affect PCA's GDScript Designer rather than the player's movement or rendering.



Figure 29. The native script-editor controls are on General in this release. They do not change Godot's separate script-editor preferences.

Setting	Use it for
Show Line Numbers	Match an error report to the relevant source line.
Automatic Indentation	Carry suitable indentation into a new line.
Autocomplete	Offer available names and calls while editing.
Show Code Minimap	Show an overview of a longer file.
Indent Using Spaces	Choose spaces rather than tabs for indentation.
Indent Size	Set the indentation width.

Keep a consistent indentation convention within a script. Changing the preference does not mean every existing file has been reformatted safely. Reopen the designer and check a short function before editing a long script.

These controls do not change the formatting of a script opened directly in Godot, and they are not a universal formatter for PCA Script, Lua, and BASIC. Use each language's own editor and syntax rules.

18. Set shared dynamic-camera following

Open **Global Settings > Dynamic Camera**. These are shared follow defaults for rooms/cameras that inherit them. They are separate from choosing a room's default camera and from its advanced camera-shot configuration.

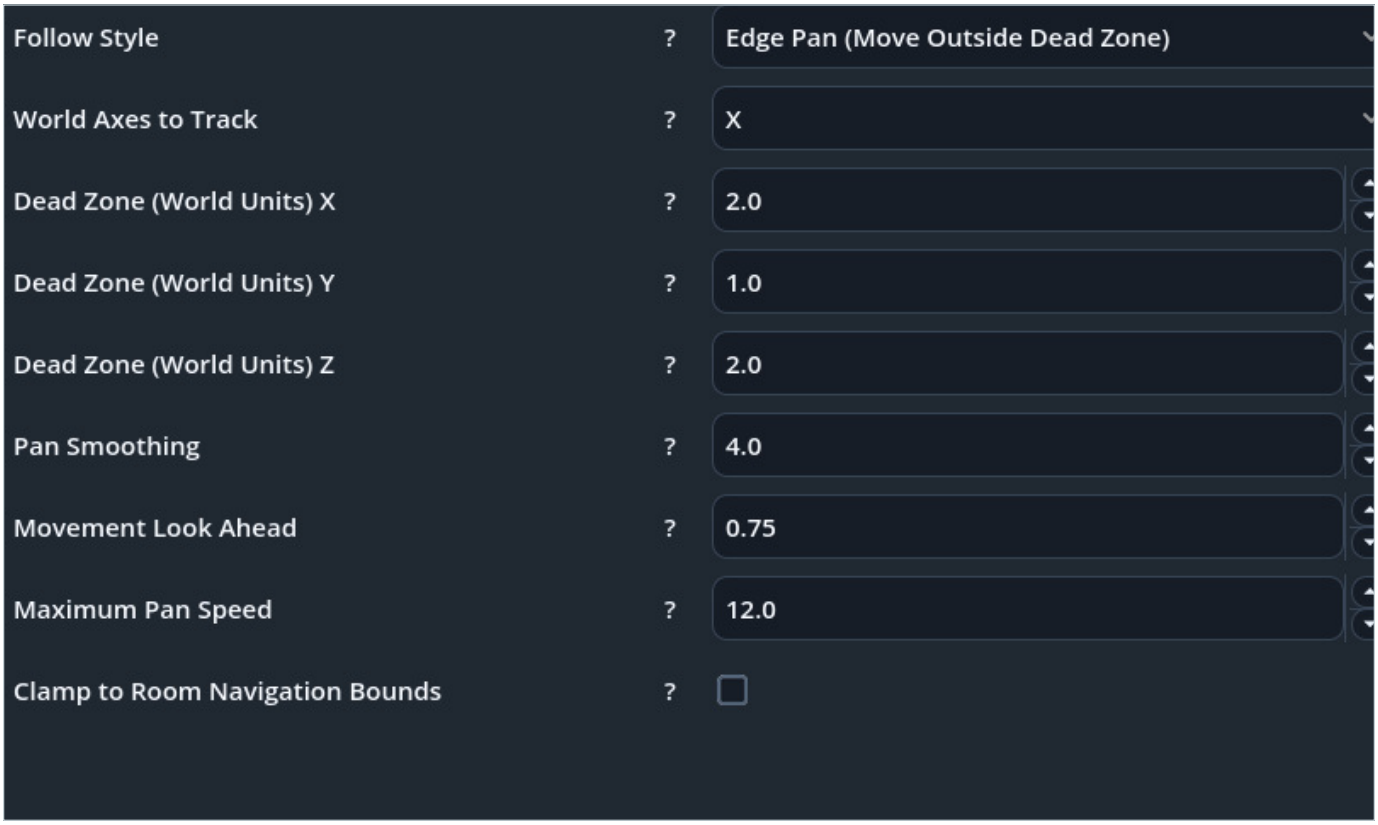

Figure 30. Enable Follow Active Player, then choose the follow style, tracked axes, and world-unit dead zone.

A first follow setup

1. Start with a working fixed room camera in Scene Designer.
2. Enable **Follow Active Player** here.
3. Choose **Edge Pan (Move Outside Dead Zone)** for a camera that stays still until the player leaves a comfortable central area, or **Smooth Follow (Continuous)** for continuous tracking.
4. Choose the **World Axes to Track**.
5. Set a sensible **Dead Zone (World Units)**.
6. Apply and walk across the room.

X and Z commonly describe travel across a 3D floor. X + Y + Z also includes height. These are world axes, not simply the horizontal and vertical pixels of a rotated camera image.

Smoothness, anticipation, and bounds



Figure 31. Look ahead, speed, and navigation-bound clamping fine-tune the camera. Room Bounds Margin appears when the clamp is enabled.

Pan Smoothing controls the response. **Movement Look Ahead** anticipates the player's direction. **Maximum Pan Speed** limits how quickly the tracking movement can travel.

Clamp to Room Navigation Bounds keeps the follow behavior within the relevant navigation extent, with **Room Bounds Margin** supplying extra space. Check the edge of the background or room geometry: a clamp that is too loose can expose unfinished scenery, while an overly tight one can prevent useful framing.

If one camera ignores the global setup, inspect the room and camera override modes. Inherit uses the parent/default; a custom setting supplies its own values; disabling dynamic tracking prevents that inherited following. Advanced Phantom Camera options belong to the camera's own setup and should be checked alongside these defaults.

A useful test route is center, left edge, right edge, near edge, far edge, then a return to center. Check the view during dialogue and cutscenes too; a puzzle camera should not fight a deliberate cinematic shot.

19. Build the global scene-shader stack

Scroll down **General** to **Default Global Scene Shader Stack**. This stack processes the rendered room/world layer. It does not automatically process your GUI or the separate screen overlay.

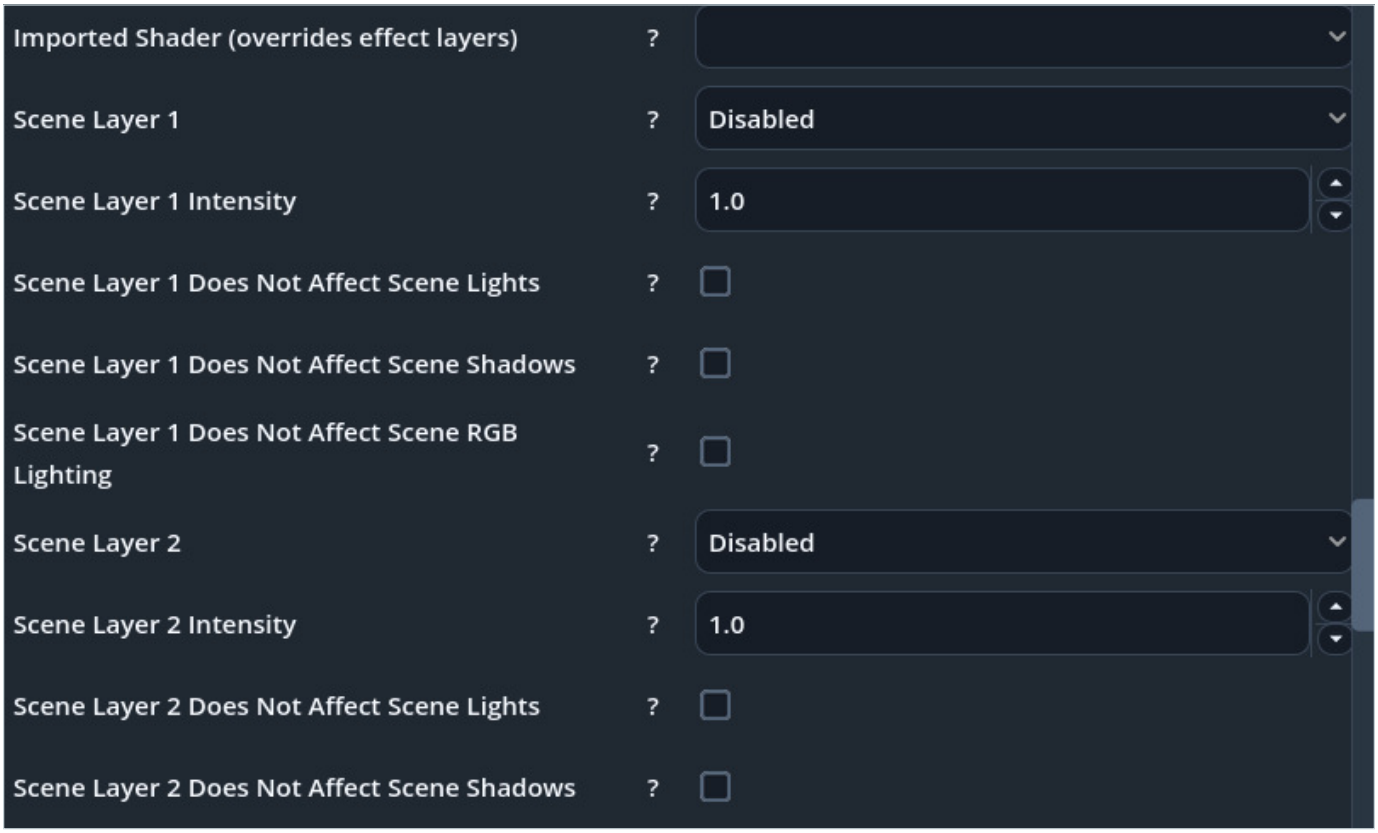

Figure 32. The shared stack has up to eight Scene Layers. An imported compatible shader overrides the preset layers.

A room can **inherit**, **replace**, or **disable** the global stack. Start with a room that inherits it when testing a new global effect. Check that room's shader override before concluding the setting did nothing.

Add one effect first

1. Leave **Imported Shader (overrides effect layers)** empty while using the presets.
2. Choose the desired preset in **Scene Layer 1**.
3. Start with a moderate **Scene Layer 1 Intensity**.
4. Leave the unused layers Disabled.
5. Apply, save, and restart the preview.
6. Check the effect on geometry, characters, readable hotspots, and the GUI boundary.

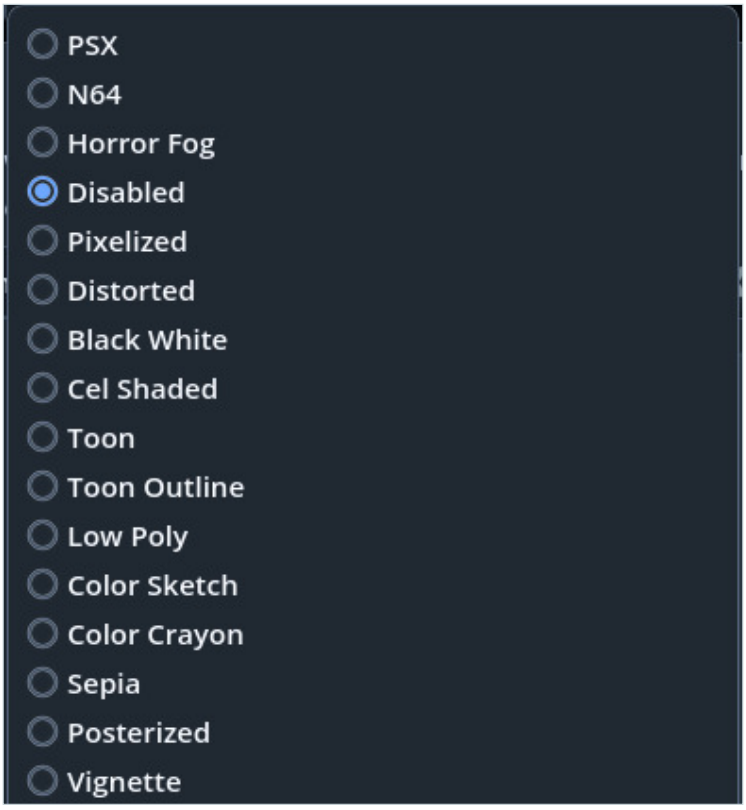

Figure 33. The native preset list includes PSX, N64, and Horror Fog alongside the other scene treatments.

Layers are arranged in order, and combining them can produce a very different result from using either alone. Start with two layers only after each works independently. An imported shader takes the dedicated override path; leaving one assigned can make changes to the preset layers appear ineffective.

For an imported global scene/overlay shader, use a compatible canvas shader asset. An object's spatial material shader is a different kind of shader. Check the asset's type and intended surface before selecting it.

PSX, N64, and horror fog as screen effects

Choose **PSX**, **N64**, or **Horror Fog** in a Scene Layer, then scroll to **PSX / N64 / Horror Fog Settings**.



Figure 34. Selecting PSX exposes its dedicated screen-effect controls. Parameter groups for other effects also share this long form.

Effect	Dedicated controls
PSX	Pixel Size, Color Bits per Channel, Dither Strength, Vertex Jitter.
N64	Pixel Size and Three-Point Filter Strength.
Horror Fog	Amount, Start Distance, End Distance, Color, Film Grain, and Vignette.

For PSX, start with a visible but restrained pixel size, then adjust color depth and dithering. Increase jitter only if that wobble is part of the intended look. For N64, compare the filtering at the actual target resolution.

For Horror Fog, keep End Distance beyond Start Distance. Tune the distances to the camera and the room scale. Fog beginning beyond the entire workshop will not help much, however atmospheric its color swatch looks.

This is the screen-effect workflow. The separate Retro Rendering group in Chapter 10 is a different route. The bundled retro material work credits Francisco Cidade (CC0); preserve the engine's included third-party credits when distributing it.

Other preset families

Family	Examples in the picker	What to watch
Color and tone	Black White, Sepia, Posterized, Invert, Warm, Cool.	Item readability and color-dependent clues.
Drawing and shading	Cel Shaded, Toon, Toon Outline, Color Sketch, Color Crayon, Hand Drawn, Stencil Drawing.	Fine details and edge strength.
Pixel/retro/display	Pixelized, CRT, VHS CRT 2, Classic Dithering, Glass Pixel.	Small text and busy patterns.

Family	Examples in the picker	What to watch
Distortion/surface	Distorted, Broken Glass, Heat Haze, Hologram, Screen Flicker.	Motion sensitivity and hotspot visibility.
Light/art treatment	Bloom, Photography, Film Grain, Metallic Toon, Aurora.	Highlights, grain, and loss of contrast.
Weather/environment	Snow, Rain, Underwater, Frostbite, Screen Fire.	Motion, density, and overlap with authored particles.

Use the controls relevant to the selected presets. **Pixel / Grain / Stroke Size**, **Distortion Amount**, **Animation Speed**, **Color / Toon / Art Levels**, **Edge / Ink Strength**, and **Effect Tint** are shared parameter groups. A control intended for one effect need not visibly change another.

Weather and specialized parameters

Snow has density, fall speed, angle, flake scale, and color. Surface accumulation adds amount, top-surface threshold, slope softness, noise scale, patchiness, and time to full coverage. Check an object that receives snow and one that deliberately does not.

Rain has density, fall speed, angle, streak scale, and color. **Underwater** has tint, tint strength, distortion, wave speed, and caustic strength. **Stencil Drawing** has ink/paper colors, darkness threshold, edge strength, and line width.

For bloom, **Bloom Lights Surrounding 3D Objects** and its color/energy/range settings are separate from merely making a bright-looking image. Keep the lighting contribution restrained and test nearby surfaces.

Each layer also has controls for not affecting scene lights, shadows, or RGB lighting paths. Test those exclusions in a simple lit scene so you can see what is being preserved. They are not commands to delete the scene's light objects.

20. Add a screen overlay and optional parallax

Open **Global Settings > Screen Overlay**. An overlay is a separate presentation layer: foreground leaves, a decorative border, a visor, or another image that sits over the world.

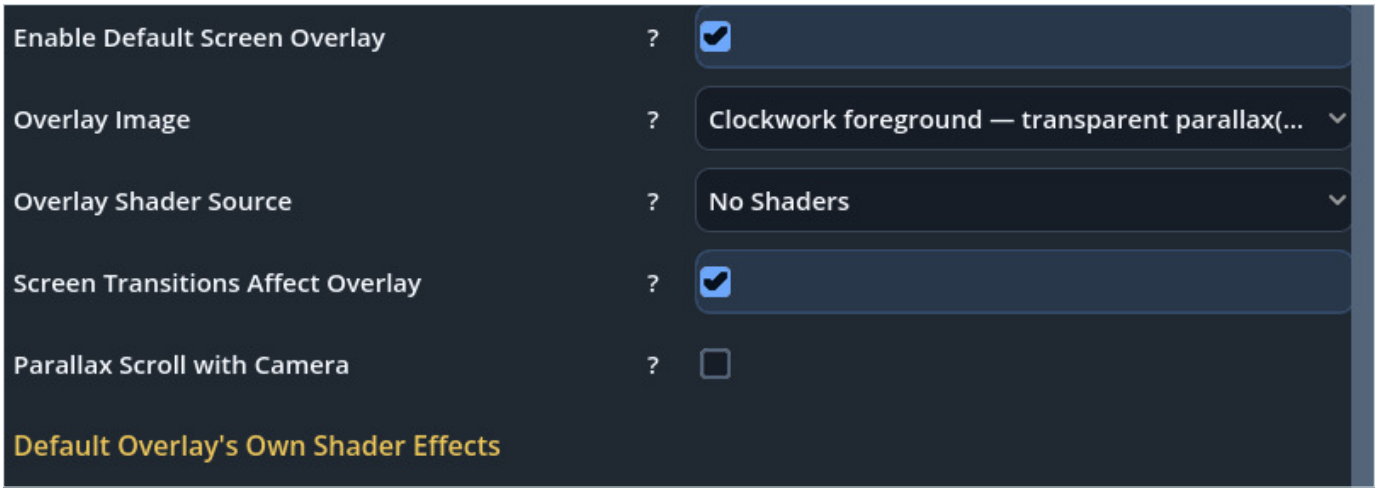

Figure 35. Enable the default overlay, choose its Image asset, and decide how it interacts with shaders and transitions.

Start with a static image

1. Import an appropriate Image asset. Use transparency when the player should see the room through it.
2. Enable **Enable Default Screen Overlay**.
3. Select **Overlay Image**.
4. Choose **Overlay Shader Source**.
5. Decide whether **Screen Transitions Affect Overlay** should be on.
6. Apply and inspect the room's edges and interactive objects.

The shader-source choices are **No Shaders**, **Use Current Room Shaders**, and **Use Overlay's Own Shaders**. The last option exposes a separate overlay-effect stack. Selecting the room-shader path is a deliberate connection; the global room stack does not otherwise process every GUI or overlay automatically.

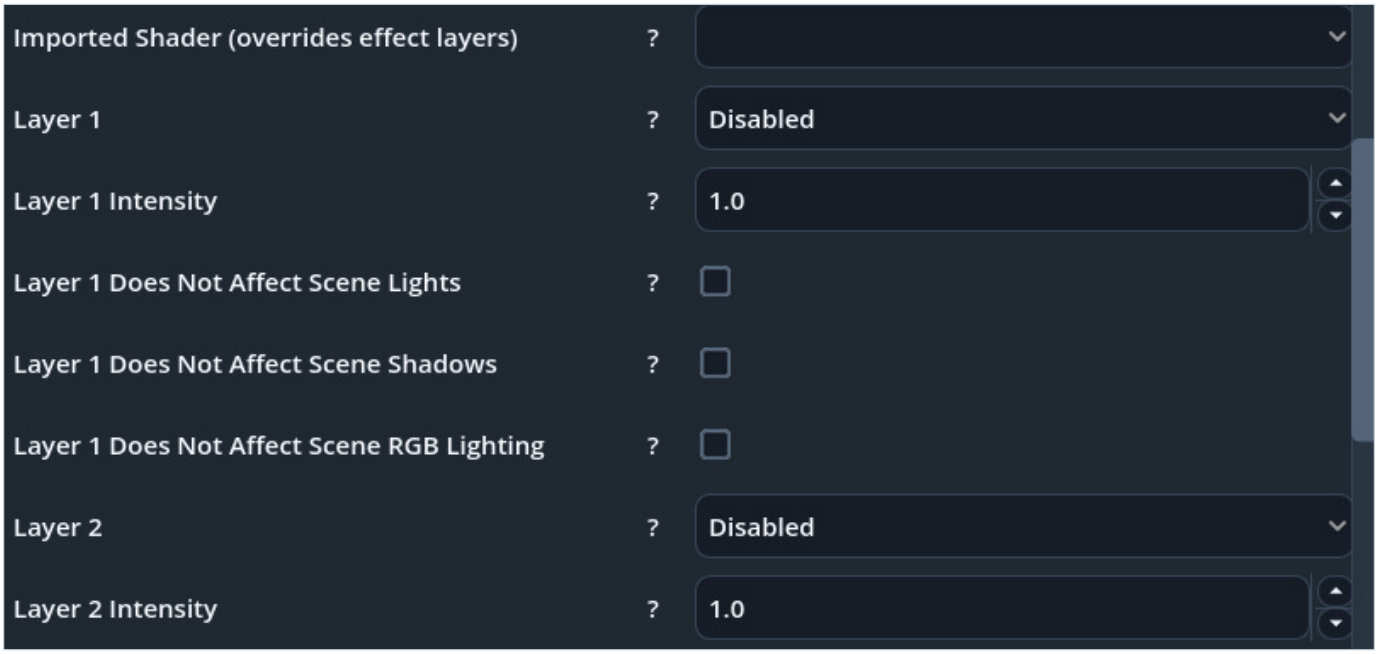

Figure 36. The overlay's own shader stack is separate from the room's stack. Its controls appear when that source is selected.

A room can inherit, disable, or supply its own overlay. Check its override if the shared image seems missing in one location.

Parallax

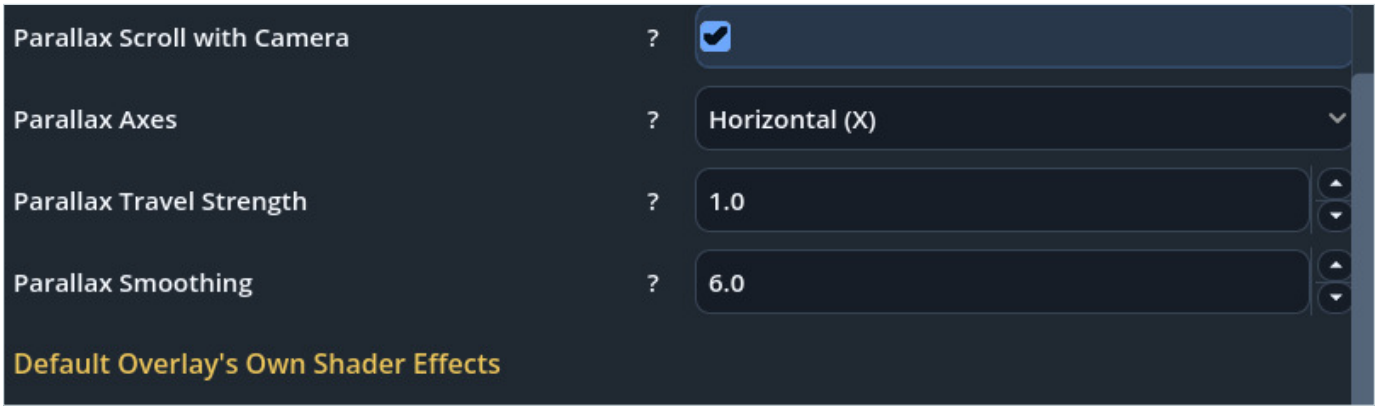

Figure 37. Parallax exposes its axis, travel strength, and smoothing controls after it is enabled.

Enable **Parallax Scroll with Camera** and choose Horizontal (X), Vertical / Depth (Z), or both. Start with low **Parallax Travel Strength**. Use **Parallax Smoothing** to control the response.

Test a camera move toward each edge. The artwork needs enough extra coverage to avoid exposing an empty strip as it travels. More strength cannot create missing pixels outside the image.

This overlay is not the same thing as a Scene Designer's independently positioned parallax planes. Use scene layers for scenery that needs authored depth/placement, and the overlay for the separate screen presentation.

21. Assign the system GUIs

Return to **General** and scroll to **Custom System Interfaces**. These selectors decide which authored GUI fills a system role.

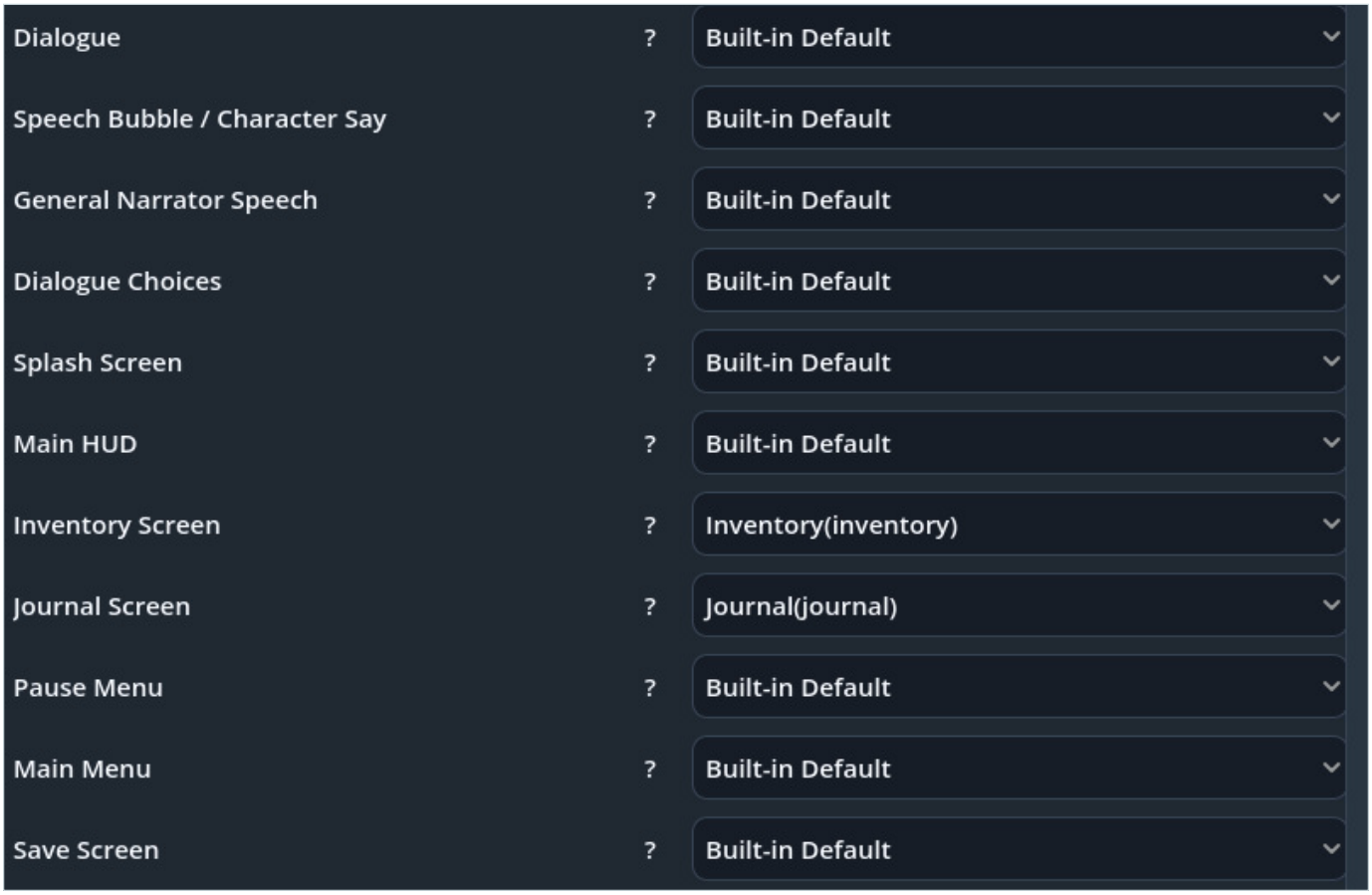

Figure 38. Each role can use an authored GUI or Built-in Default. Main HUD also has a deliberate No HUD choice.

Role	Purpose
Dialogue	The main dialogue presentation.
Speech Bubble / Character Say	Character-speech presentation.
General Narrator Speech	General messages/narration.
Dialogue Choices	The choice-selection interface.
Splash Screen	An authored startup GUI splash.
Main HUD	The ordinary gameplay interface, or No HUD.
Inventory Screen	The inventory interface.

Role	Purpose
Journal Screen	The journal interface.
Pause Menu	The paused-game menu.
Main Menu	The opening/main menu.
Save Screen / Load Screen	Save-slot interfaces.
Options Screen	Player settings controls.

Create and test the GUI first, then select it here. A selector assignment does not automatically turn every decorative button in that GUI into a working system control.

Important group bindings

The speech text group is `@talkdialogue`. Portrait groups are `@playerpic` and `@NPCpic`. Choice groups are `@choice1` through `@choice20`, with `@choicescrollup` and `@choicescrolldown` for additional choices. Preserve the exact case of those names. Clicking the correctly bound talk text can continue a spoken line.

Use the GUI Designer's system/group features for the rest of the interface behavior. A button labeled Save is still only a button until it has the relevant binding/action.

A playable character can also select personal HUD, inventory, journal, or dialogue GUIs. Those more specific choices can affect the interface you actually see. Check Character Designer when one character uses a different layout.

The two splash systems

The **Splash Screen** GUI assignment and **Splash Duration (seconds)** belong to the authored GUI splash. Project Settings' ordered Startup Splashes have their own media records and durations.

If both are configured, the ordered media splashes run and the authored GUI splash can also appear before the normal startup flow. Test the whole opening so you do not accidentally show two versions of the same studio logo.

22. Choose the player's starting options

Open **Global Settings > Player Options**. These values establish the starting preferences. A released standalone game can later load that player's saved choices instead.



Figure 39. Master, Music, SFX, and Voice volumes are separate defaults. Values run from 0 to 1.

Audio

Use **Master Volume** for the overall level and Music/SFX/Voice for their respective channels. Zero silences a channel. Start with a clear voice and comfortable music level, then test a line over music and a loud effect.

Changing a default volume does not retune every source recording. A clipped or extremely quiet recording may need editing at the asset level. Keep the dialogue recording and typing-blip sample on their intended channels.

Text, language, display, and accessibility

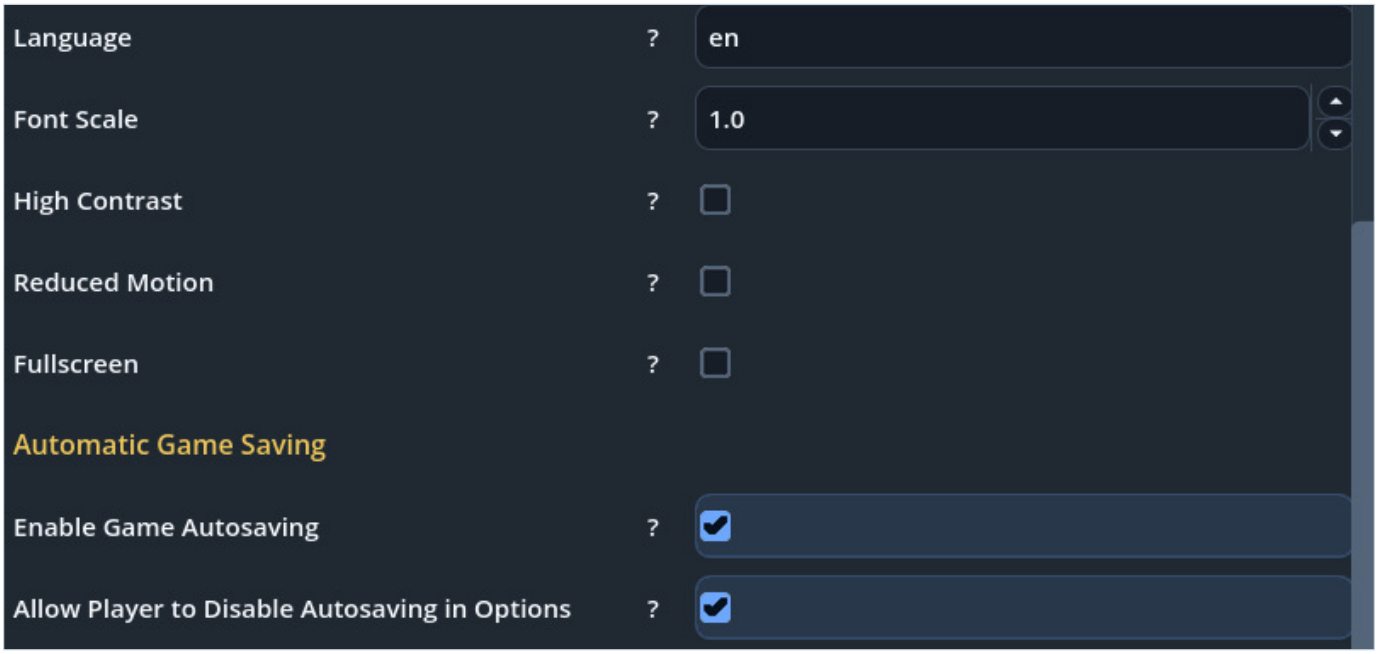

Figure 40. Language, font scale, accessibility preferences, and fullscreen are player defaults. Check them in an actual game window as well as the editor preview.

Option	What to use/check
Language	A game-language code configured through Game Languages & Translations.
Font Scale	The supported UI's text scaling preference. Test wrapping and button sizes.
UI Font / Dialogue Font	Font records created in the Font designer; separate general and dialogue choices.
High Contrast	The supported text treatment. It does not automatically redesign every custom GUI.
Reduced Motion	Supported camera/effect/GUI paths can reduce motion. Check authored/custom effects too.
Fullscreen	The standalone game-window preference. Embedded preview does not switch the whole editor to fullscreen.
Text Speed / Subtitles	Stored text-presentation preferences; see the current-build note below.

In this build, the line's own typewriter configuration remains the reliable way to set its reveal speed. The Text Speed preference is not a universal override of every authored text path. Likewise, the Subtitles checkbox should not be assumed to hide every ordinary dialogue label or custom caption. Test the specific presentation or make your authored UI honor the preference.

A larger Font Scale can expose a narrow button or portrait column. Test long translated text too. An explicit font/size on an authored GUI element may need its own adjustment.

Preview defaults versus a player's saved options



Figure 41. The running game exposes its player options. In this test, a runtime action changed Master Volume to 0.40; the authored font-scale default is 1.20.

The embedded preview initializes from the project defaults. A standalone game can load an existing saved options file, which represents that player's previous choice. This explains why the same build may open at a different volume or language on a machine that has run it before.

When checking a newly authored default, use the embedded preview or restore the standalone game's default options through the intended interface. Keep existing player saves/options safe while testing. Do not treat a remembered preference as proof that the project failed to save.

23. Set saving rules, screenshots, and the main menu

Saving has several related controls. Keep the author's permission to autosave separate from the player's preference to use it.

Automatic Game Saving

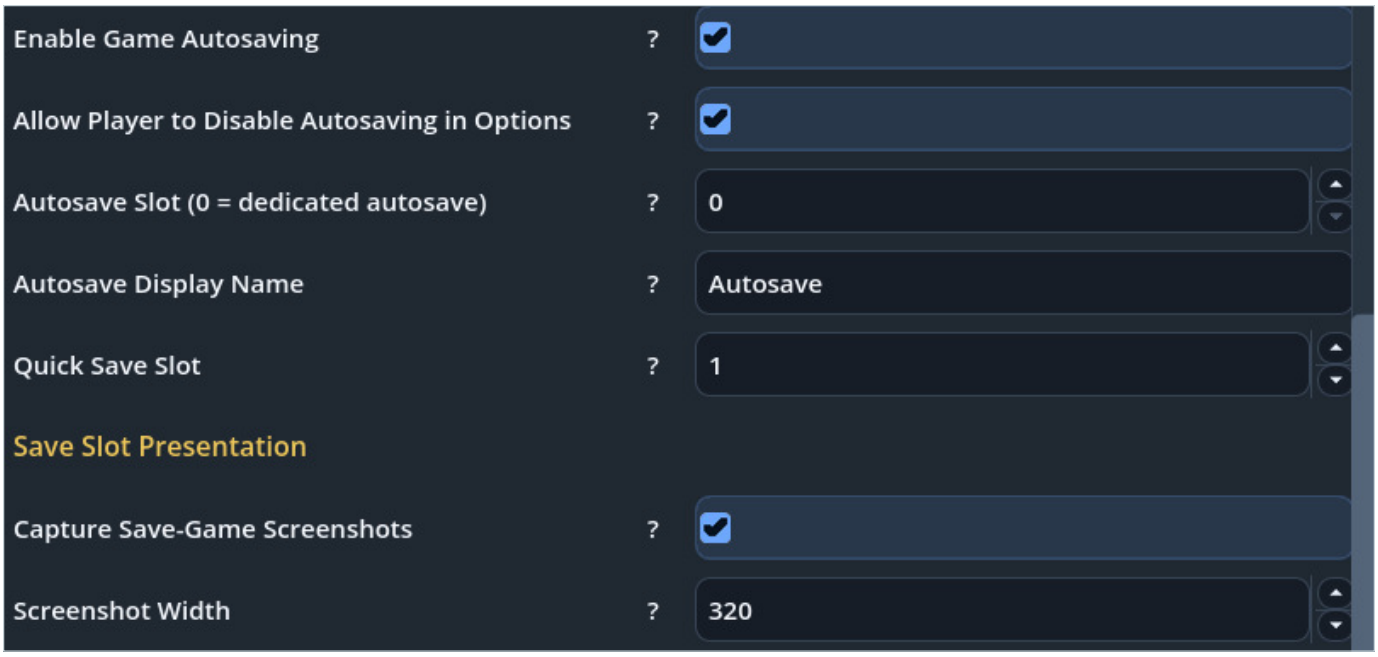

Figure 42. Enable Game Autosaving is the author-level choice. Allow Player to Disable Autosaving and Player Default set the player's role in that decision.

1. Set **Enable Game Autosaving** for the intended game behavior.
2. Decide whether **Allow Player to Disable Autosaving in Options** should be on.
3. Choose **Autosave Slot**, normally 0 for the dedicated autosave.
4. Enter **Autosave Display Name**.
5. Choose a **Quick Save Slot** that will not surprise the player.
6. Check **Player Default: Automatic Game Saving**.
7. Apply, save, and test a new session, a manual save, a quick save, and loading.

The author-level autosave switch must permit autosaving. The player's enabled preference also matters. The older Project Settings Autosave flag is a fallback for data without the explicit save-system choice; it is not the editor's own project-autosave feature.

Save Slots lives on Screen Overlay in this release

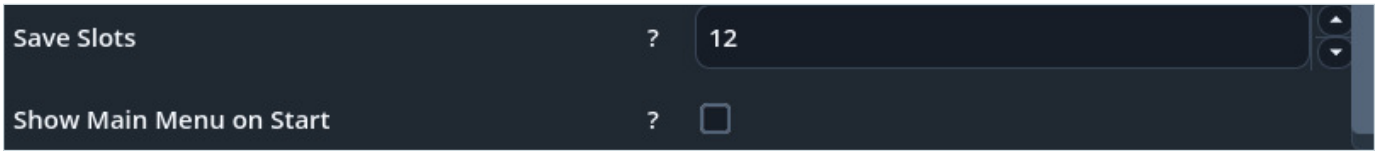

Figure 43. Save Slots and Show Main Menu on Start are at the bottom of Screen Overlay, below its visual controls.

Set **Save Slots** there for the number of regular slots. Its form supports 1 to 100. Slot 0 has the dedicated autosave meaning in the autosave setting. Keep autosave and quick-save slot choices within the intended arrangement; otherwise an automatic/quick save can overwrite a slot the player thought was a separate manual save.

Show Main Menu on Start also lives here. It requests the opening main menu; the actual GUI assignment is **General > Custom System Interfaces > Main Menu**. If an authored menu is assigned, make sure its New Game/Continue/Load/Options actions are working.

The full startup can include ordered media splashes, the authored GUI splash, then the opening main menu. Beginning a new game proceeds into the configured startup action. Test that route and a load-game route separately.

Save Slot Presentation



Figure 44. Save screenshots, dimensions, the default name template, and the player autosave preference are grouped on Player Options.

Enable **Capture Save-Game Screenshots** for thumbnails. Choose **Screenshot Width** and **Screenshot Height**; 320 x 180 is a modest 16:9 starting point. A large thumbnail costs more storage and is not necessary merely because the game runs at a large resolution.

Default Save Name Template can use a pattern such as `Save {slot}`. **Autosave Display Name** supplies the automatic slot's label separately. Check the actual Save/Load widget after changing them.

A Save/Load GUI still needs the proper widget/bindings and slot layout. These defaults do not redraw the widget's corner radii or replace its background pictures. Those are GUI Designer appearance settings.

24. Map gamepads and generic joysticks

Open **Global Settings > Controllers**. Start with the actual device connected.

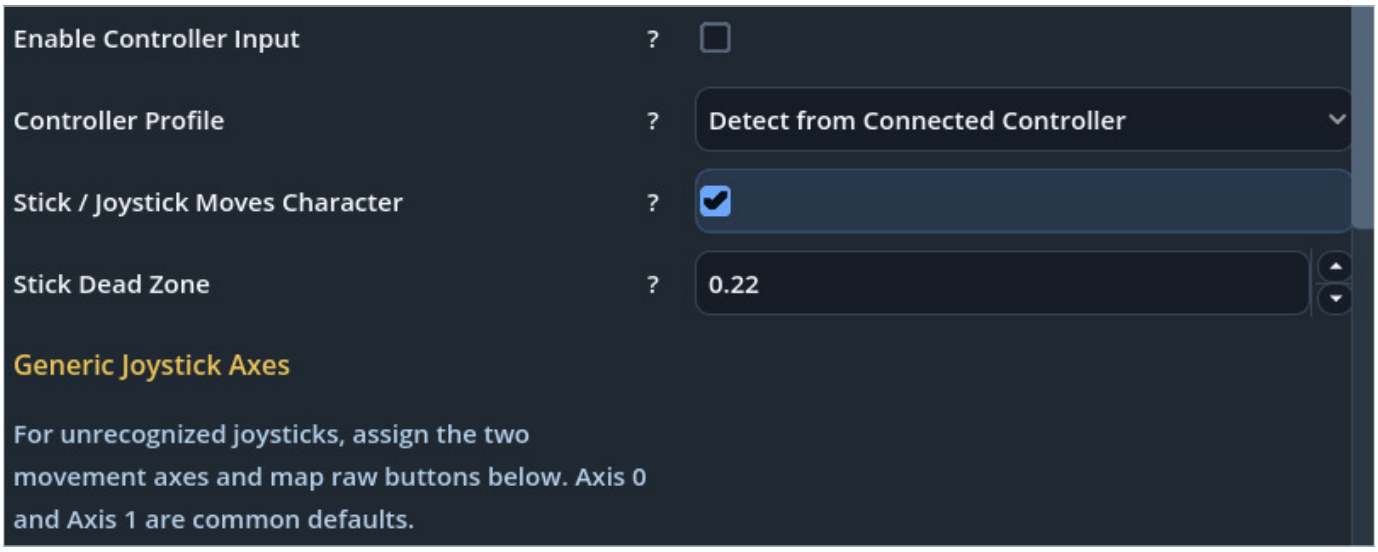

Figure 45. Enable Controller Input, choose detection/profile behavior, and tune the stick dead zone before changing many button assignments.

First connection

1. Enable **Enable Controller Input**.
2. Start with **Detect from Connected Controller**, or choose the intended profile explicitly.
3. Enable **Stick / Joystick Moves Character** when the stick should control movement.
4. Leave a modest **Stick Dead Zone** and test the stick at rest.
5. Test movement and one easy button action, such as Pause / Menu.
6. Apply, save, and test a dialogue and inventory as well as walking.

The profiles shown are Xbox, PS5 DualSense, Nintendo Switch 2, and Generic Joystick. Detected names depend on the device mapping. If a controller is classified unexpectedly, try the explicit matching profile and check the actual buttons.

A dead zone that is too low can allow drift. Too high can make a gentle stick movement feel unresponsive. Change it in small steps with the device at rest and moving slowly.

Generic axes

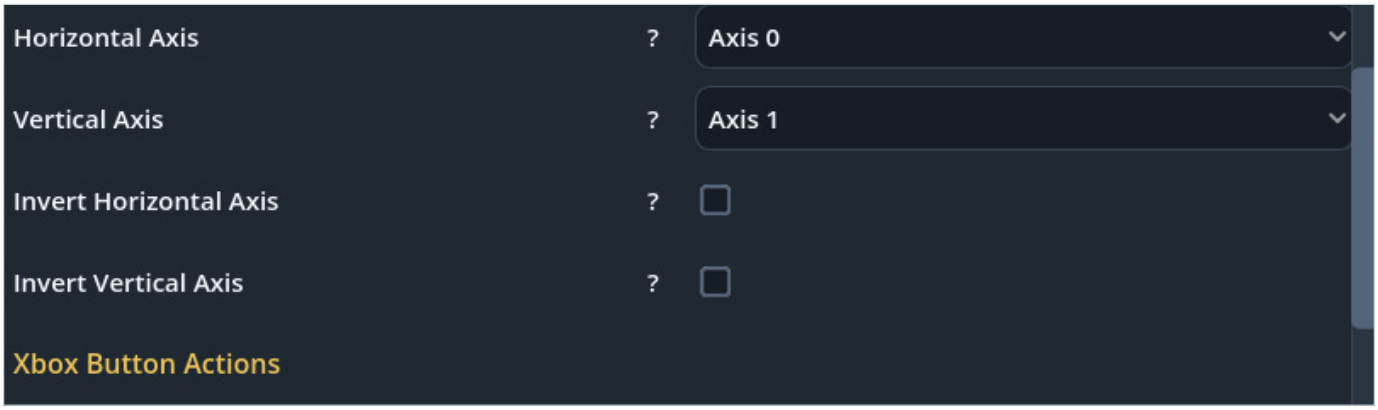

Figure 46. Generic Joystick Axes identifies the horizontal/vertical movement axes and optional inversions.

For an unrecognized joystick, choose **Horizontal Axis** and **Vertical Axis**. Axis 0 and Axis 1 are common starting points, not a guarantee for every device. Use the two inversion checkboxes if the observed direction is reversed.

Move one axis at a time while testing. Do not compensate for a wrong axis assignment by rotating a character model or changing the camera's world axes.

Button actions

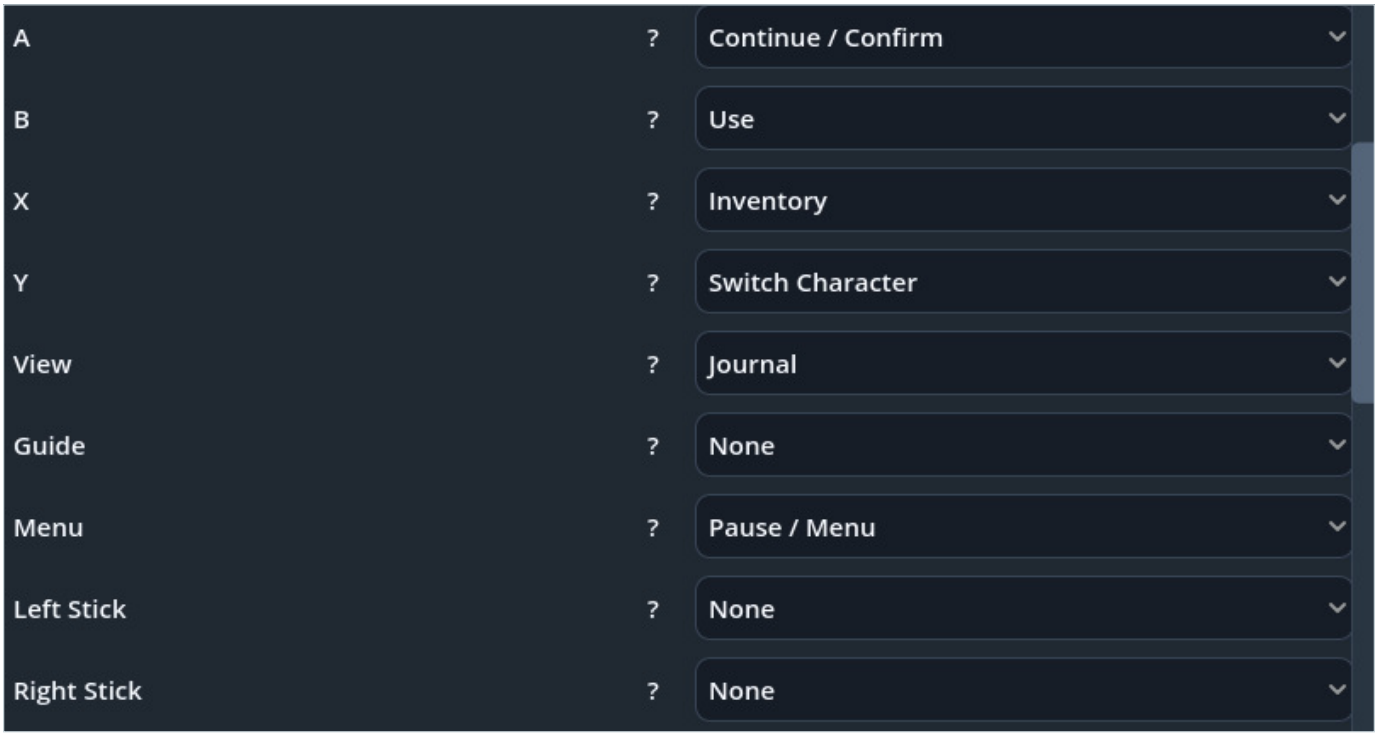

Figure 47. Each profile has its own button-action section. Other profiles remain available farther down the page.

Use the named profile's section for its buttons. Generic Joystick exposes raw Button 0 through Button 15. Assign actions deliberately and leave unused buttons as None.

Available actions include Interact, Look, Interaction Coin, Next/Previous Hotspot, Cycle Verb, Continue / Confirm, menus, inventory, journal, character switching, skipping cinematics, verb shortcuts, parser focus, and directional movement.

The underlying face-button positions are consistent, but the printed letters differ between controller families. Check the prompts on the actual device instead of copying an Xbox letter onto a differently labeled pad.

For proximity-style play, also review Adventure & Cursor's profile, proximity radius, cursor adaptation, and hints. The controller mapping receives input; those settings determine the intended interaction style.

25. Set touch gestures, first-person sticks, and player camera navigation

Mobile Controls

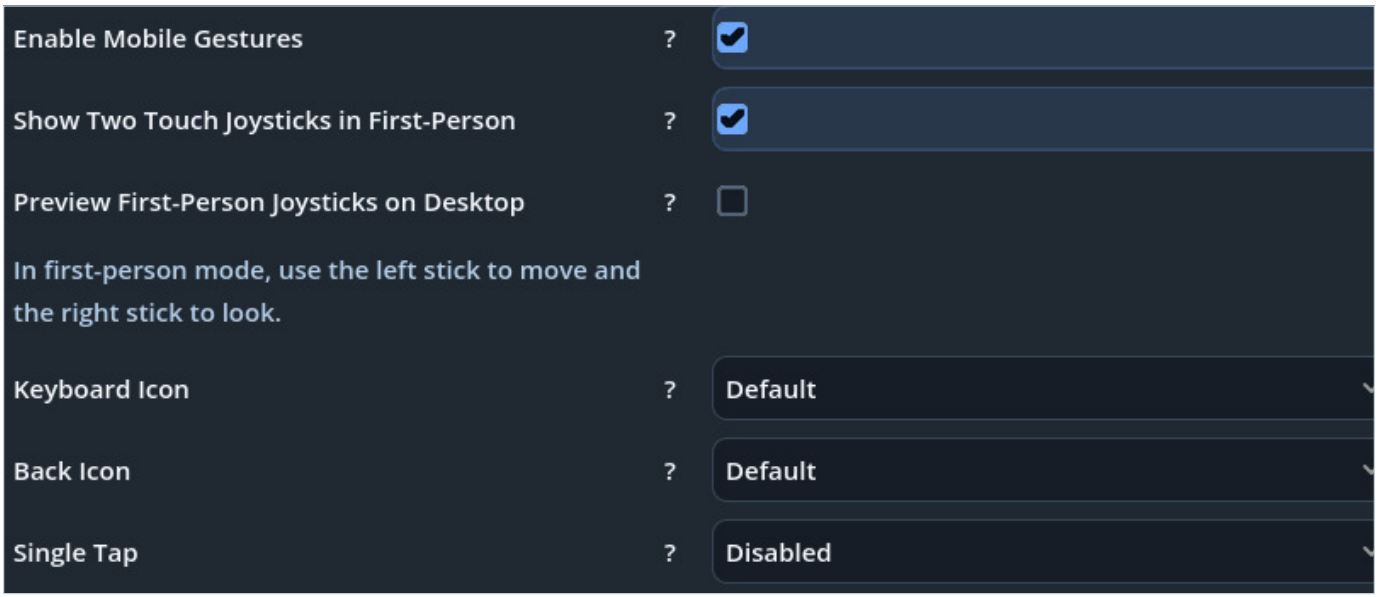

Figure 48. The mobile page controls gestures, first-person touch sticks, and optional keyboard/back icons.

Enable **Enable Mobile Gestures** for the gesture system. **Show Two Touch Joysticks in First-Person** supplies left-move/right-look sticks in that view. **Preview First-Person Joysticks on Desktop** is useful when inspecting their layout, but final touch testing still belongs on the target device.

Choose **Keyboard Icon** and **Back Icon** from Image assets when you want custom artwork, or keep their defaults. Check transparent margins and touch-target size rather than judging only the artwork's visible pixels.

Assign the gestures

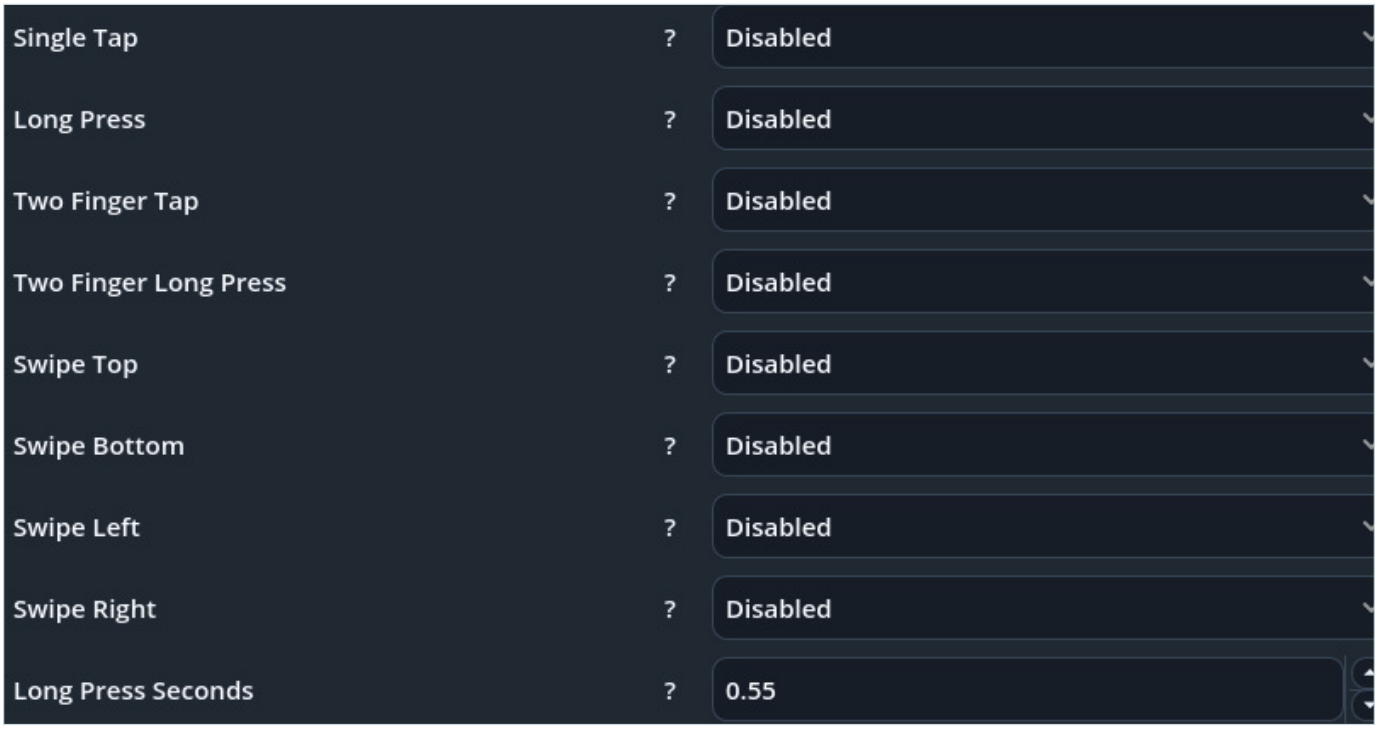

Figure 49. Single/long/two-finger taps and edge swipes can each receive an action. Long Press Seconds controls the hold threshold.

The available gestures are Single Tap, Long Press, Two Finger Tap, Two Finger Long Press, Swipe Top, Swipe Bottom, Swipe Left, and Swipe Right.

They can be assigned None, Interact, Move, inventory/journal/pause, parser focus, next character, zoom, or pan as appropriate. Keep an ordinary tap easy to predict. Reserve two-finger gestures for secondary actions and give important actions a visible interface route too.

Long Press Seconds decides how long the hold must last. About half a second is a useful starting point. Test a deliberate long press and an ordinary tap with slight finger movement so the two do not feel interchangeable.

If touch should use Drag to Move, also set that device's movement profile in Adventure & Cursor. A gesture assignment and an automatic movement-profile choice are separate parts of the setup.

Scene Navigation is the player's view, not the editor's view



Figure 50. Scene Navigation enables runtime pinch/wheel zoom and two-finger pan. It does not enlarge walkable navigation.

Enable **Pinch / Wheel Zoom** and set **Closest Zoom**, **Farthest Zoom**, and **Zoom Step**. Keep the closest limit below the farthest one. Enable **Two Finger Pan** and set **Maximum Pan from Center** when panning is intended.

These affect player navigation of the view. They are not the editor viewport's orbit/zoom controls, and they do not enlarge a 2.5D ground area. If the character cannot walk beyond a boundary, inspect the room's navigation and colliders.

Test zoom at both limits, then pan near the limit while zoomed in and out. Keep the player able to find the character and see enough scenery to understand where to go next. Check how this interacts with an authored following camera before allowing unrestricted pan in the whole game.

26. Configure VR defaults

Open **Global Settings > VR Controls** for a project intended to use a compatible VR build/runtime. The enable checkbox is part of that setup; it does not turn an ordinary desktop window into a finished headset build.



Figure 51. VR basics include locomotion, inventory placement, profile, movement speed, teleport distance, and snap turning.

Movement and interface

Field	Choice to make
Enable VR in VR Builds	Enable the configured VR path.
Locomotion	Teleport, Continuous, or Both.
Inventory Interface	Front or Wrist placement.
Continuous Movement Speed	The speed used by continuous locomotion.
Controller Profile	Choose the appropriate listed device profile.
Teleport Distance	The permitted teleport reach.
Snap Turn Degrees	The size of a snap-turn step.

Begin with comfortable, restrained movement and turning. Test the inventory at the intended scale and distance, not only as a flat rectangle on the editor monitor. Teleport destinations still need to make sense in the room's navigation/collision setup.

Generic buttons and device overrides

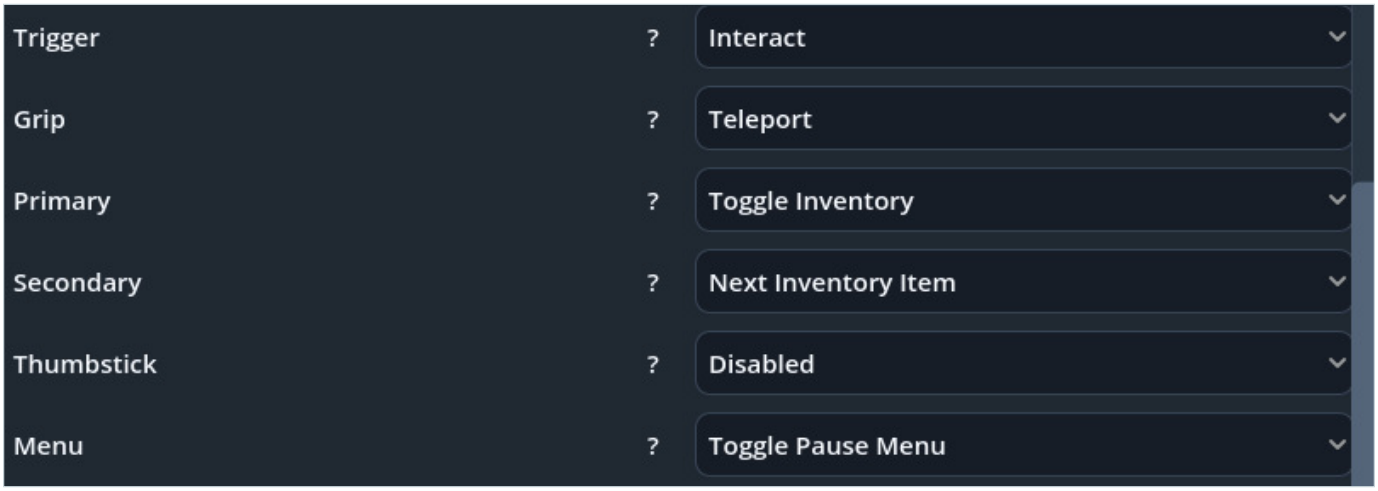

Figure 52. Generic VR Button Actions sets the shared Trigger, Grip, Primary, Secondary, Thumbstick, and Menu behavior.

The shared button actions include interaction, teleporting, inventory controls, pause, character switching, and None. Assign the common arrangement first.



Figure 53. Per-device sections can inherit the shared action or override it. Check the correct section for the active profile.

The listed override groups include Quest 3, Steamframe, PSVR 2, and Generic. **Inherit** keeps the shared button choice. A specific action replaces it for that profile.

OpenXR uses the active headset/controller bindings. Test the complete arrangement on each intended headset. A profile selector is not a substitute for the platform's export support, SDKs, or licensing; console/headset-specific exports can have separate requirements.

Use a short, uncluttered test room before adding a cinematic scene. Check movement, turn, interact, inventory, pause, and returning from the pause interface.

27. Make a working baseline and change settings during play

The designer establishes the starting values. Shared actions and scripts can make supported changes during a running game.

A manageable baseline

Use your existing small scene. No special practice project is needed.

1. Project Settings: choose a title, 1280 x 720, an explicitly placed playable character, and Start in Player Character Location.
2. Choose Context interaction and a deliberate shared/per-character inventory arrangement.
3. Global Settings: choose a renderer appropriate to the machine and begin with a restrained quality setting.
4. Use one movement profile and keep automatic input switching off for the first test.
5. Leave cinematic grading, extra shader layers, and the overlay off until the room is readable and playable.
6. Use Built-in Default interfaces initially unless a working authored GUI is already available.
7. Set comfortable audio/font defaults and deliberate saving rules.
8. Apply, save, Validate, and use Play / Restart.
9. Add one optional feature, then repeat the test.

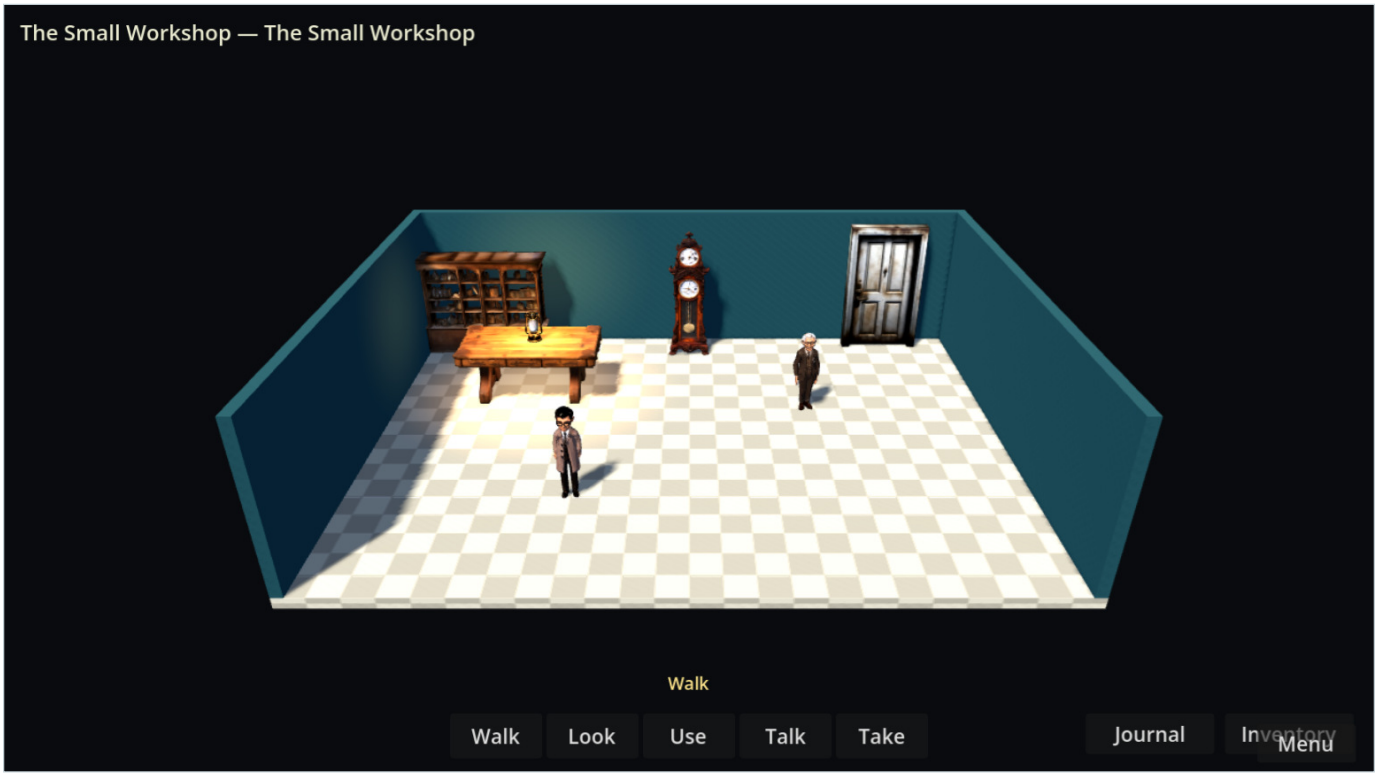

Figure 54. The test starts the actual workshop game through the configured player/startup path. Extra presentation systems can be added after this baseline works.

The pictured verification started with Point and Click, Context interaction, Master Volume 0.65, and Font Scale 1.20. The running game then accepted Direct Movement, Verb Single Click, and Master Volume 0.40 through the shared runtime controls. Those are separate choices, not one giant “control mode.”

Action lists and visual scripts

Action	Typical fields
Set Movement Profile	Movement Profile.
Set Interaction Mode	Interaction Mode.
Automatic Input Switching	Enabled.
Set Input Device	Automatic, Mouse and Keyboard, Gamepad, or Touch.
Proximity Interaction	Enabled; shared action ID `SetProximityInteractionEnabled`.
Set Control Setting	One profile setting and a correctly typed value.
Get Control State	A variable in which to store the current state.
Change Game Setting	A setting name and the new value.

Action	Typical fields
Change Project Property	A project-data property path and the new value.

Use the dedicated control actions for profile/interaction/input changes. They update the live control system. For example, a cutscene can switch to a particular interaction arrangement and restore the previous one afterward.

Not every stored field is a safe instant runtime change. A rendering-backend switch requires a game/application restart. Use a quality change or a supported presentation/control action when that is the effect you actually need.

GDScript example

Put this inside the entry function provided by the script editor. Keep its template wrapper and parameters. These values match the verified runtime change described above.

```
await SetMovementProfile({"profile": "direct"})
await SetInteractionMode({"mode": "verb-single-click"})
await ChangeGameSetting({
    "setting": "masterVolume", "value": 0.4
})
```

For the existing settings API, `GetSetting("renderQuality")` reads the value and `SetSetting("renderQuality", "performance")` requests a supported quality change. Check return values where the call provides one. Setting a value in a running session does not write your authored project file back to disk.

PCA Script

```
function run()
    SetMovementProfile({"profile": "direct"})
    SetInteractionMode({"mode": "verb-single-click"})
    ChangeGameSetting({"setting": "masterVolume", "value": 0.4})
end
```

BASIC

```
FUNCTION Run()
    AWAIT SetMovementProfile({"profile": "direct"})
    AWAIT SetInteractionMode({"mode": "verb-single-click"})
    AWAIT ChangeGameSetting({"setting": "masterVolume", "value": 0.4})
END FUNCTION
```

Lua

```
local Script = {}
function Script.run()
    await(SetMovementProfile({profile = "direct"}))
    await(SetInteractionMode({mode = "verb-single-click"}))
    await(ChangeGameSetting({setting = "masterVolume", value = 0.4}))
end
return Script
```

The action names and property names are shared across these systems. Their language syntax and waiting conventions differ. Use actual identifiers/values from the selector when changing an enumerated mode.

28. Find a setting, troubleshoot it, and finish the project

Where to look

Job	Location
Title, author, description	Project > General.
GUI click/default footstep	Project > General > Default Game Sounds.
Base resolution	Project > General > Game Resolution.
Starting player and startup action	Project > General > On Game Startup.
Interaction/inventory mode and pixel-perfect 2D	Project > General > Gameplay Defaults.
Ordered image/video startup cards	Project > Startup Splashes.
Executable/package names, data-pack name, icon	Project > Packaging.
Translations and CSV	Game Languages & Translations above Project Settings.
Renderer, quality, grading, enhanced lighting	Global > General.
GUI/overlay design-size independence, hover names	Global > General.
Native GDScript editor options	Global > General, farther down.

Job	Location
Global room screen effects and their parameters	Global > General > Default Global Scene Shader Stack.
System-GUI assignments and GUI splash duration	Global > General > Custom System Interfaces.
Walk/approach, input/movement profiles, keyboard, dialogue, cursors	Global > Adventure & Cursor.
Shared player-follow camera	Global > Dynamic Camera.
Global overlay and its own shaders/parallax	Global > Screen Overlay.
Save Slots and opening main-menu switch	Global > Screen Overlay, at the bottom.
Player volume/language/font/accessibility defaults	Global > Player Options.
Autosaving, quick-save slot, screenshots, save-name template	Global > Player Options.
Gamepad/joystick input	Global > Controllers.
Touch gestures and first-person sticks	Global > Mobile Controls.
Player view zoom and pan	Global > Scene Navigation.
VR locomotion and mappings	Global > VR Controls.
Editor theme, editor language, editor UI size	Editor toolbar/preferences.
Local export tools and target setup	Build Game / Build Tools.

When a setting seems ineffective

Symptom	Check in this order
A value returns to the old choice	Apply the correct form, save the project, then check for a discarded draft.
An effect works in one room only	Room override mode, camera/object-specific settings, then global defaults.
Forward+ effect is absent	Selected profile, actual Preview renderer, required application restart, and effect support.

Symptom	Check in this order
New default volume/language differs in a standalone build	Existing saved player options; compare with fresh embedded preview defaults.
Global walk speed affects only some characters	Character custom-speed setting and scene overrides.
A door still blocks close approach	Physical collider/blocking, then approach distance; event volume is separate.
A shader-layer change does nothing	Assigned imported-shader override, room inheritance, selected preset and relevant parameter.
Overlay is missing or shifts unexpectedly	Global enable/image, room override, parallax coverage, and design-size independence.
No custom dialogue portrait appears	System GUI assignment, exact picture group tags, speaker/portrait setup.
Gamepad mode does not receive input	Controllers enabled, detected/explicit profile, hardware mapping, then movement profile.
Touch action conflicts with walking	Gesture assignment, long-press/drag threshold, and active touch movement profile.
First-person sticks are missing in desktop editing	First-person setup and the explicit desktop-preview checkbox.
Fullscreen does not fill the editor	Embedded preview deliberately does not change the whole editor window. Test a standalone game.
Autosave is disabled unexpectedly	Author-level permission, player default/current preference, and the explicit save-system settings.
Startup cards/script do not run in a room test	Use Play / Restart for the full startup sequence.
The default camera does not follow	Global enable, room/camera inheritance, tracked axes, dead zone, bounds, and advanced camera choices.

Raw JSON (Advanced)

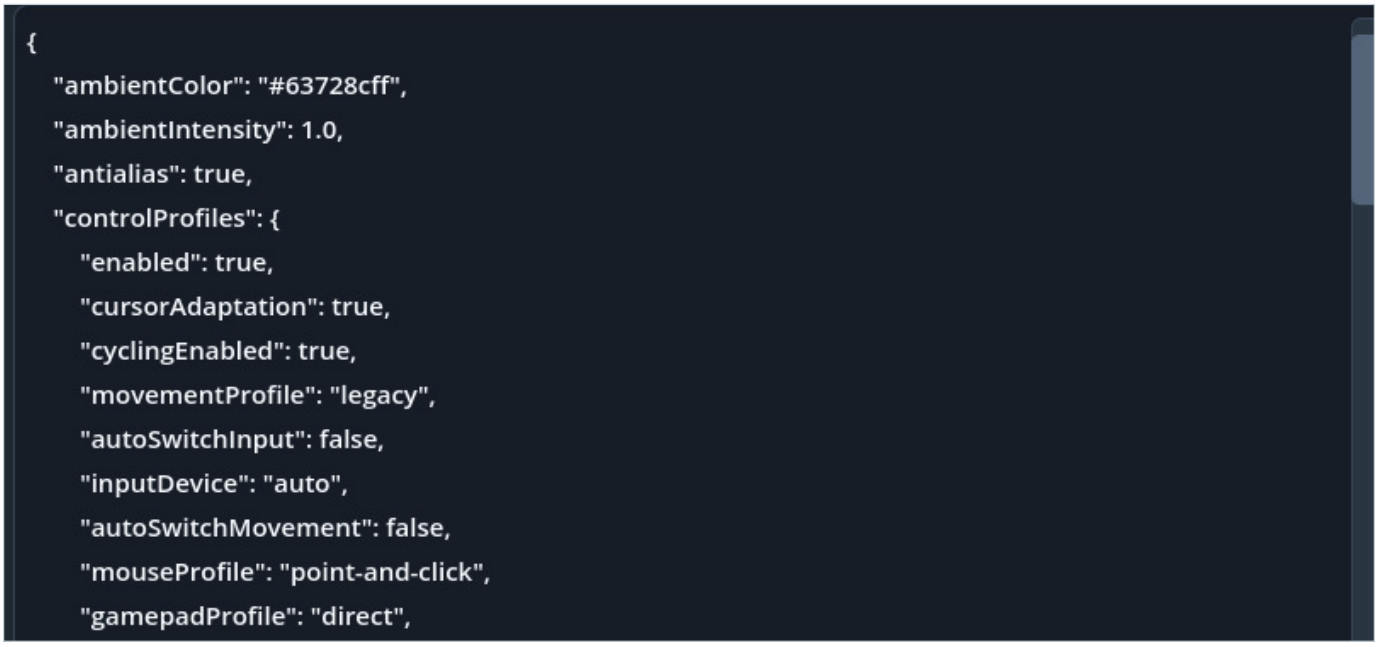

Figure 55. Raw JSON shows the selected settings record. Use it for a deliberate advanced edit, then validate the result.

Project Settings' JSON page edits the project metadata record. Global Settings' JSON page edits the settings record. Do not wrap another `meta` or `settings` object around values just because that wrapper exists in the complete project file.

Use normal fields first. Before an advanced edit, save a working version, change one known property, apply, and revisit its normal field. Keep stable asset/character/GUI IDs intact. A valid JSON document can still contain a reference to something that does not exist.

Before distributing the game

1. Check the intended title, base resolution, icon, and platform filename stems.
2. Run the full startup, including skipping splashes and choosing New Game from the menu if used.
3. Test the starting player and a character switch where supported.
4. Test each intended input device and its hints.
5. Check a room that inherits defaults and one that deliberately overrides them.
6. Test long dialogue, translated text, inventory, pause, options, save, and load.
7. Check both author defaults and a returning player's saved preferences.
8. Validate and resolve missing references.
9. Save the project and build the intended target.
10. Launch the complete output on the target machine with its supporting files present.