PCA 3D ADVENTURE STUDIO

The Scene Designer Part 2

Models, animation and actions



Give the room something to do.

Bring in real 3D art, animate the characters, and build a small interaction with action lists, a camera close-up and a repair flag. Then add a trigger, local lighting, a region effect and a small particle emitter.

For version 0.86.3.4 - Windows, macOS and Linux

Actual editor screenshots. No practice files required.

How to use this guide

This is the follow-up to the first Scene Designer guide. Start with your own saved workshop or another usable 3D room. Follow the core art and character steps first, then add the interaction. The lighting, region and particle chapters are useful optional additions.

Part	What you will build
Chapters 1-6	Imported furniture, sensible placement and fitted collision.
Chapters 7-10	An actual animated player and NPC, with mapped gameplay states.
Chapters 11-17	A clock interaction with a message, close camera, gesture, sound and game variable.
Chapters 18-19	A conditional door response and a one-time entry hint.
Chapters 20-22	A warm point light, local region effect and a small flame.
Chapters 23-24 and reference	A complete play test, troubleshooting and the main settings.

The artwork in the pictures comes from the engine's bundled Clockmaker content. You can use your own model files. Their scale, origin and animation names may differ, so the lesson explains what to check instead of assuming every file is identical.

Screenshots are cropped where a close view is more useful than the whole desktop. Read the selected record, tab and field labels. Your theme and panel widths may look different.

Apply, save, then test. Keep the plain working version while adding one new behaviour at a time. No companion project, asset pack or practice files are supplied.

Contents

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

1. Turn the room into a small adventure . . . 4
2. Know which record you are editing . . . 6
3. Import a proper 3D prop . . . 7
4. Bring the model into the scene . . . 9
5. Put the imported art on the floor . . . 10
6. Match the collider to the furniture . . . 13
7. Import and inspect an animated character . . . 16
8. Give the player the animated model . . . 17
9. Tell PCA 3D which clip means walk . . . 19
10. Upgrade the NPC and test movement . . . 23
11. Give the clock a Look response . . . 25
12. Add a close-up camera . . . 26
13. Create a variable for the repair . . . 28
14. Build the clock's Use action list . . . 29
15. Let the character approach the clock . . . 35
16. Make the NPC speak . . . 36
17. Test the complete repair sequence . . . 38
18. Make the door check the repair flag . . . 41
19. Add an automatic entry hint . . . 45
20. Give the lantern a real light . . . 47
21. Add a local region effect . . . 48
22. Add a small flame effect . . . 51
23. Test the scene as a whole . . . 55
24. Fix the common problems . . . 58
R1. The important settings at a glance . . . 60

1. Turn the room into a small adventure

The first Scene Designer guide built a room from boxes and used simple character placeholders. This time we will furnish that room with imported models, give the player real animation, and make the objects do something useful.

The little scene has a straightforward job. A clock has stopped. The player examines it, makes an adjustment, hears a chime, and sees the camera return to the room. A game variable remembers the repair. An archive door checks that variable before it opens. A small entry trigger, a lantern light and a local region effect finish the lesson.

These instructions follow PCA 3D Adventure Studio **0.86.3.4**. They apply to Windows, macOS and Linux. The screenshots are from the running editor and its game preview, with some pictures cropped so that the settings remain readable.

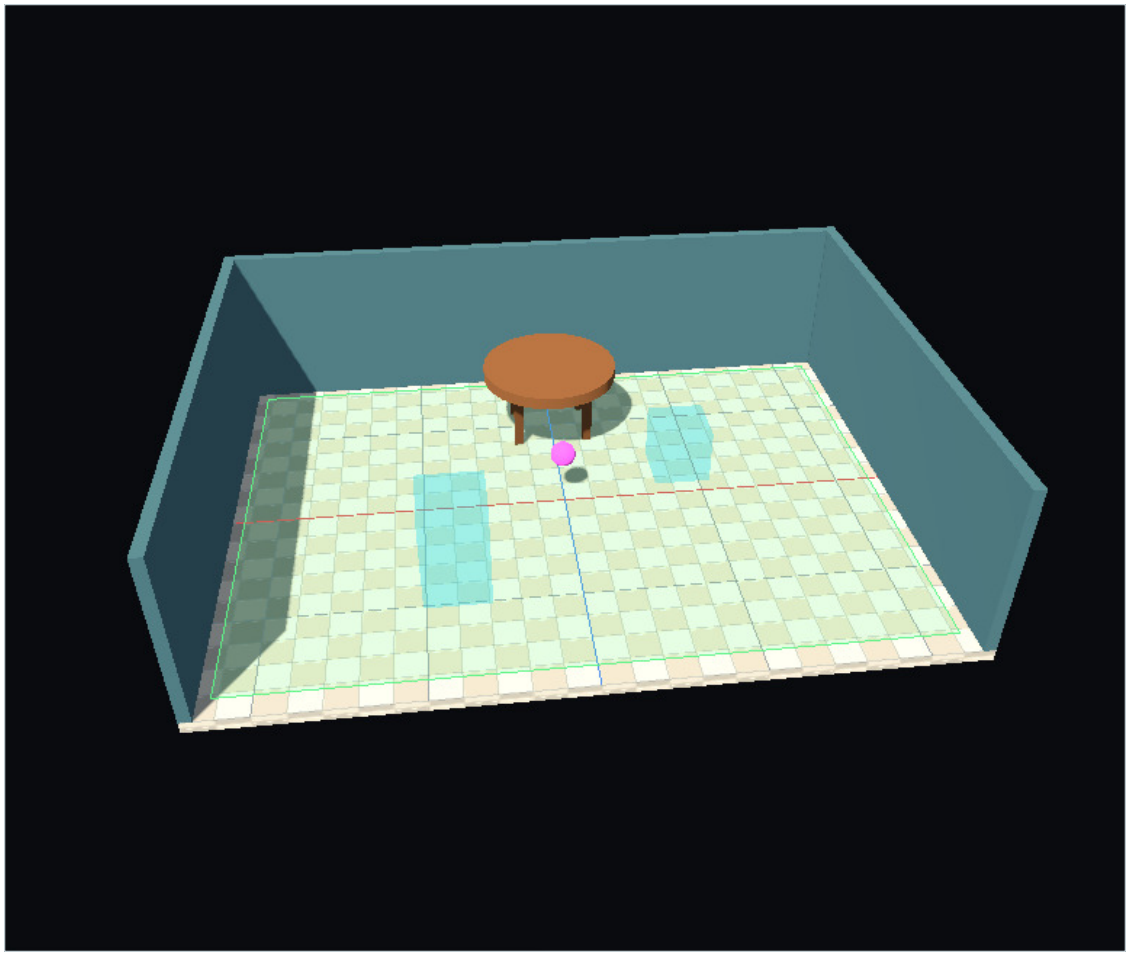

Figure 1. Our starting room: the floor, walls, navigation and room camera from the first guide.

What you need before starting

Open the workshop you made in the first guide, or use another saved 3D room with a floor, a working navigation area, a player and a default camera. Save a copy of your own project before changing its art or behaviour.

For the art steps, have one static 3D prop and one animated character available as model files. A GLB with its materials embedded is convenient. The character needs actual animation clips; a model standing in a T-pose is not, by itself, a walking character.

The illustrations use the engine's bundled Clockmaker models. They are examples of the workflow, rather than a separate art pack supplied with this book. You can inspect the bundled models in a project created from the Clockmaker template. To import into your own workshop, choose model files you already have permission to use. Their clip names and dimensions may differ from the examples.

No practice project, ZIP or asset download accompanies this guide. You can follow the same steps with your own models. Where a measurement depends on the pictured model, the text says so.

The names used in the lesson

Record	Example name and ID	Its job
Room	The Small Workshop(workshop)	Contains the scene.
Player	Player(player)	The character controlled by the player.
NPC	Mara(mara)	The workshop keeper from the first guide.
Clock model asset	Workshop Clock(clock_model)	Reusable source artwork.
Clock scene object	Workshop Clock(clock)	The placed clock, with interaction actions.
Room camera	Room View(wide)	The ordinary gameplay view.
Detail camera	Clock Close-up(clock_close)	A closer shot during the repair.
Variable	Clock Repaired(clock_fixed)	Remembers whether the repair has happened.

The name is the readable label. The ID is what a reference points to. Give both sensible values before adding a lot of actions.

2. Know which record you are editing

Most problems in this lesson come from editing the right-looking thing in the wrong place. The following distinction is worth getting straight before importing anything.

You want to change...	Edit this
Which file supplies a model	Its Asset record in the Asset Designer.
Where one table stands in a room	The placed table in Scene Objects.
A character's reusable model and animation states	The Character definition in the Character Designer.
Where that character starts in this room	The character's room instance in Scene Objects.
What happens when the clock is used	The clock object's Verb Actions.
Which shot opens the room	The room's Default Camera.

An imported asset is not automatically placed in the room. A model placed as an ordinary Asset object does not automatically become a playable Character. An animation clip does not automatically acquire the gameplay job called `walk`.

Keep checking the Inspector heading. **Entity - Workshop Clock** is the placed object. **Scene - The Small Workshop** is the entire room. In the Character Designer, check the selected character at the top before changing states.

Use **Apply Changes** for a completed set of Inspector fields, **Apply Asset Changes** in the Asset Designer, and **Validate and Apply State** in the Animation State Editor. Then use **Save Project**. Those buttons commit different kinds of record.

3. Import a proper 3D prop

We will use a grandfather clock as the main interactive prop. A cabinet, machine or other static model would work just as well.

1. Open the **Asset Designer**.
2. Click **Add New**.
3. Enter Asset ID `clock_model` and Name **Workshop Clock**.
4. Set **Kind** to **Model**.
5. Beside **Asset File**, click **Browse...** and choose the model file.
6. Let the import finish. Do not immediately start a second import while it is still copying files or preparing the preview.
7. Click **Apply Asset Changes**, then Save Project.

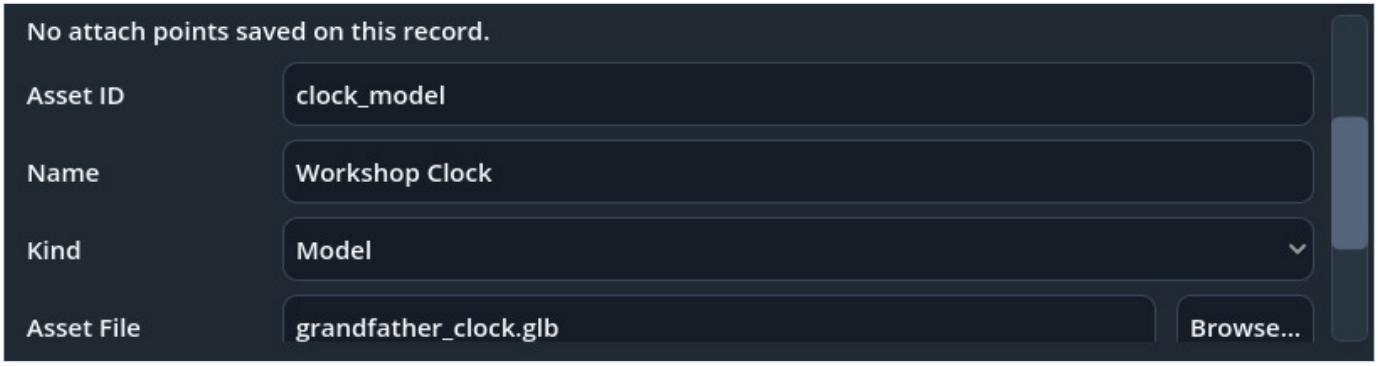

Figure 2. The source model is an Asset record. Kind must be Model for a 3D prop.

If you are learning inside a Clockmaker template project, its models are already imported. Select the existing asset and use its own ID in later selectors; importing a second copy is unnecessary. The workshop pictures use new example asset names. A model ID identifies a record in the current project, rather than locating a file in another project.

Keep the model's companion files together

A GLB often contains its geometry and textures in one file. A glTF package may have a separate BIN file and texture folder. An OBJ may refer to an MTL file and images. Keep those companion files with the model when you browse for it, so the importer can find them.

The importer copies supported dependencies into the engine's asset area. Still, check the result. A successful file selection is not proof that every material is correct.

Inspect the preview before placement

Use **Frame Model** if the object is difficult to see. Turn Auto Rotate off when you want to examine one side. Front, Side and Top help you spot an unexpected orientation. You can orbit, pan and zoom without changing the placed room object.

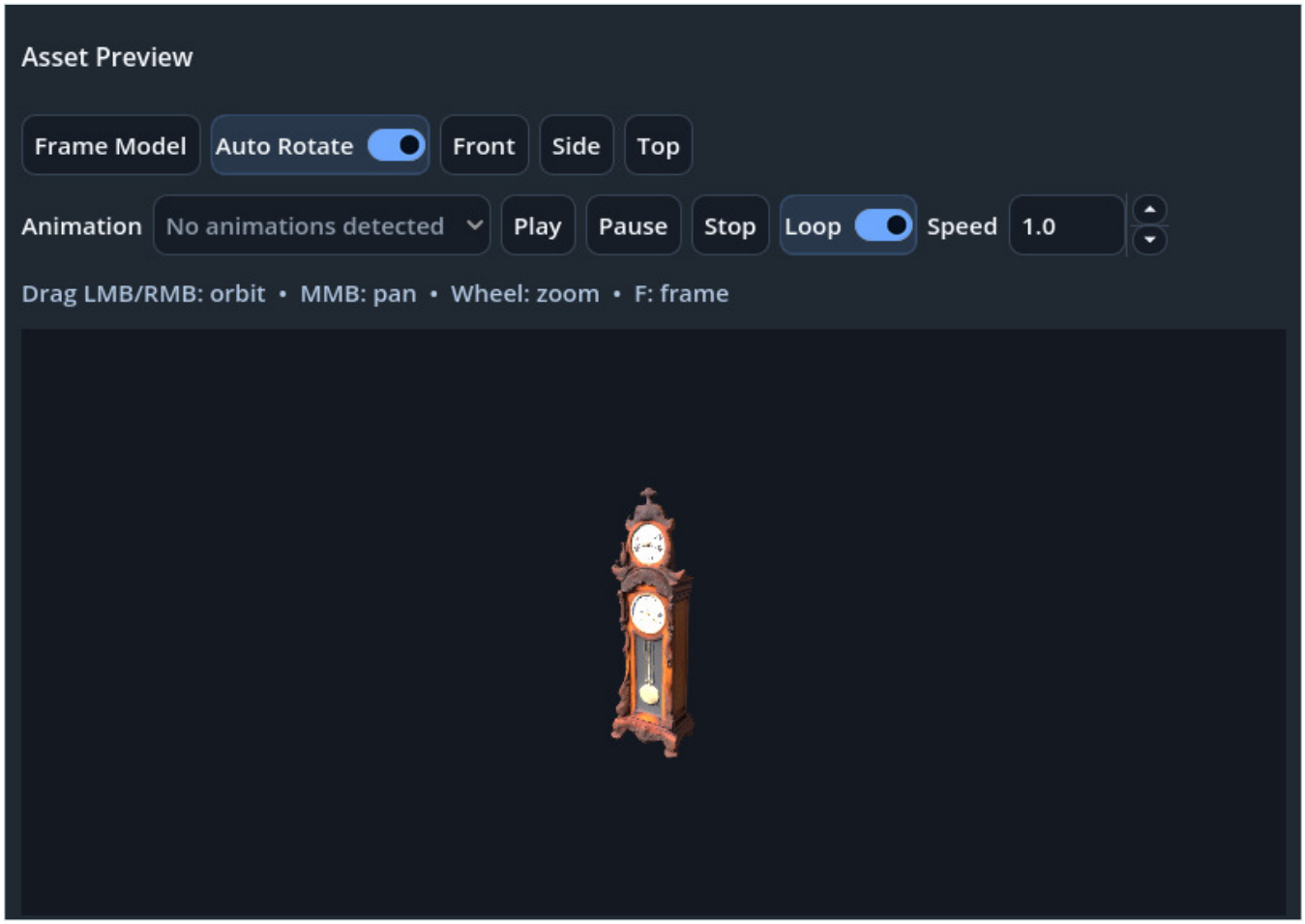

Figure 3. The clock in Asset Preview. A static prop correctly reports that it has no animation clips.

Check its shape, materials and orientation. If it is already white or missing textures here, fix the asset before spending time on room lighting. If it looks correct here but wrong after placement, inspect the placed object's material overrides, colour tint and lighting.

4. Bring the model into the scene

Return to the Scene Designer. There are two useful ways to work: place another object, or change the artwork of an existing object.

Place a new model

1. Show the **Asset Placer** panel if it is hidden. The Panels menu controls the editor panels.
2. Choose the imported model in the asset list and place it in the 3D view. Dragging a supported model asset into the scene is another placement route.
3. Select the newly placed object under **Assets** in Scene Objects.
4. Give the object ID `clock` and Name **Workshop Clock**.
5. Open Visual Adjustments and enter its final position, rotation and scale.
6. Apply and save.

You can also use **Add to Scene > Imported 3D Asset**, then choose the model in the selected object's Model / Image / Sprite Sheet field. The important result is a placed object of Type Asset referencing the correct model record.

The source asset can be `clock_model` while this particular placed object is `clock`. Several placed objects may share one source asset. Give each placement its own ID if actions need to distinguish them.

Replace the generated workbench

The first guide used a generated round table. To keep its place in the object list while replacing its art:

1. Select **Workbench** in Scene Objects.
2. In General, change **Type** to **Asset**.
3. Set **Model / Image / Sprite Sheet** to your imported table model.
4. Apply Changes.
5. Open Visual Adjustments and adjust the model's position and scale.
6. Check blocking and the collider again, then save.

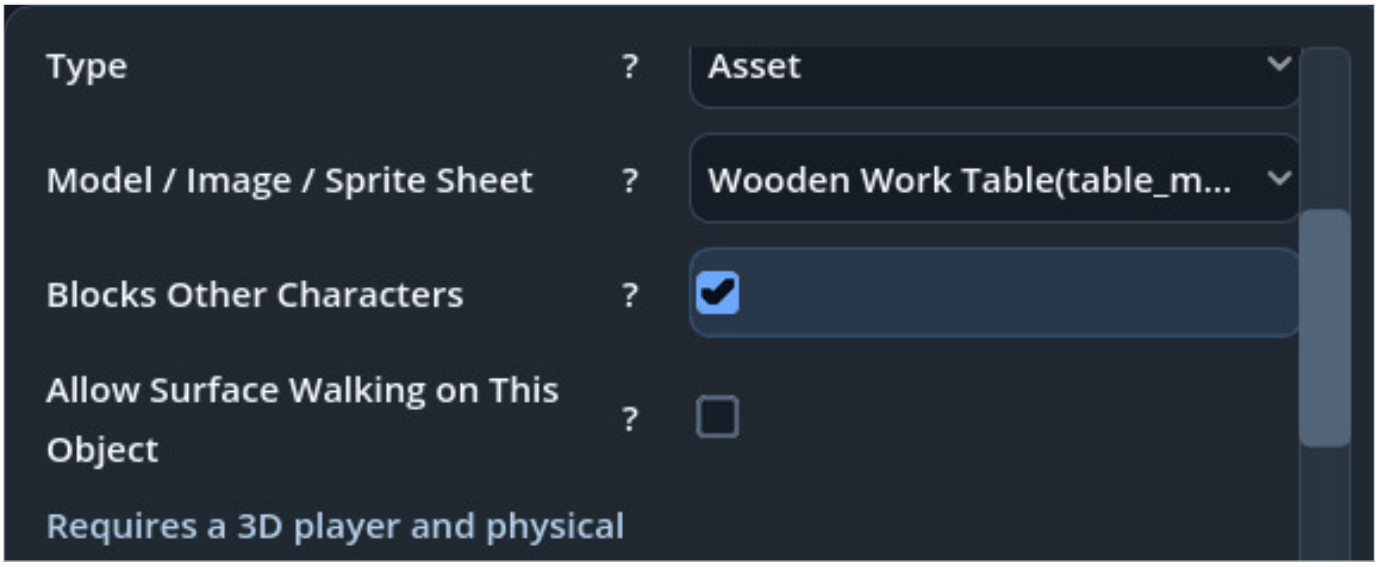

Figure 4. Workbench now references a model asset. Its scene-object identity is separate from the source model's identity.

A primitive's old Size values do not resize an imported mesh in the same way. Use the imported object's **3D Scale** values. Keep all three scale values equal when you want to preserve the model's proportions.

5. Put the imported art on the floor

Imported models do not all share the same origin. Some are measured from their feet. Others are centred halfway up their height. That is why a model can appear half-buried when its Y position is zero.

Fit the workbench

In the pictured workbench, the model's origin is roughly halfway up the table. At uniform Scale **1.4**, Position Y **0.57** puts its feet on the floor.

Set Position to **-3.2, 0.57, -1.8**, Rotation to 0, 0, 0, and Scale to **1.4, 1.4, 1.4** for this particular model.



Figure 5. The workbench's placement and scale. This Y value compensates for this model's centred origin.

For your own table, inspect it from Front or Right view and adjust Y until its feet meet the floor. Do not copy 0.57 blindly onto a model whose origin is already at its feet.

Place the clock and other props

For the pictured clock, use Position **0, 1.4, -3.7**, Rotation 0, 0, 0, and uniform Scale **1.4**. Its centred origin needs that Y offset too.



Figure 6. The clock's scene position. Its model is lifted so that its base sits on the floor.

These measurements reproduce the pictured furniture layout. They are starting points for these exact models, not universal import settings.

Object	Position X, Y, Z	Uniform Scale	Rotation Y
Workbench	-3.2, 0.57, -1.8	1.4	0
Clock	0, 1.4, -3.7	1.4	0
Bookcase	-4.5, 1.12, -3.7	1.4	1.5708
Archive Door	4.3, 1.405, -4	1.4	0
Bench Lantern	-3.1, 1.49, -1.8	0.35	0

The lantern rests on the table, so its placement uses the table-top height rather than floor height. The bookcase needs a quarter turn because its original orientation differs. The ordinary 3D Rotation fields use radians: 1.5708 is about 90 degrees.

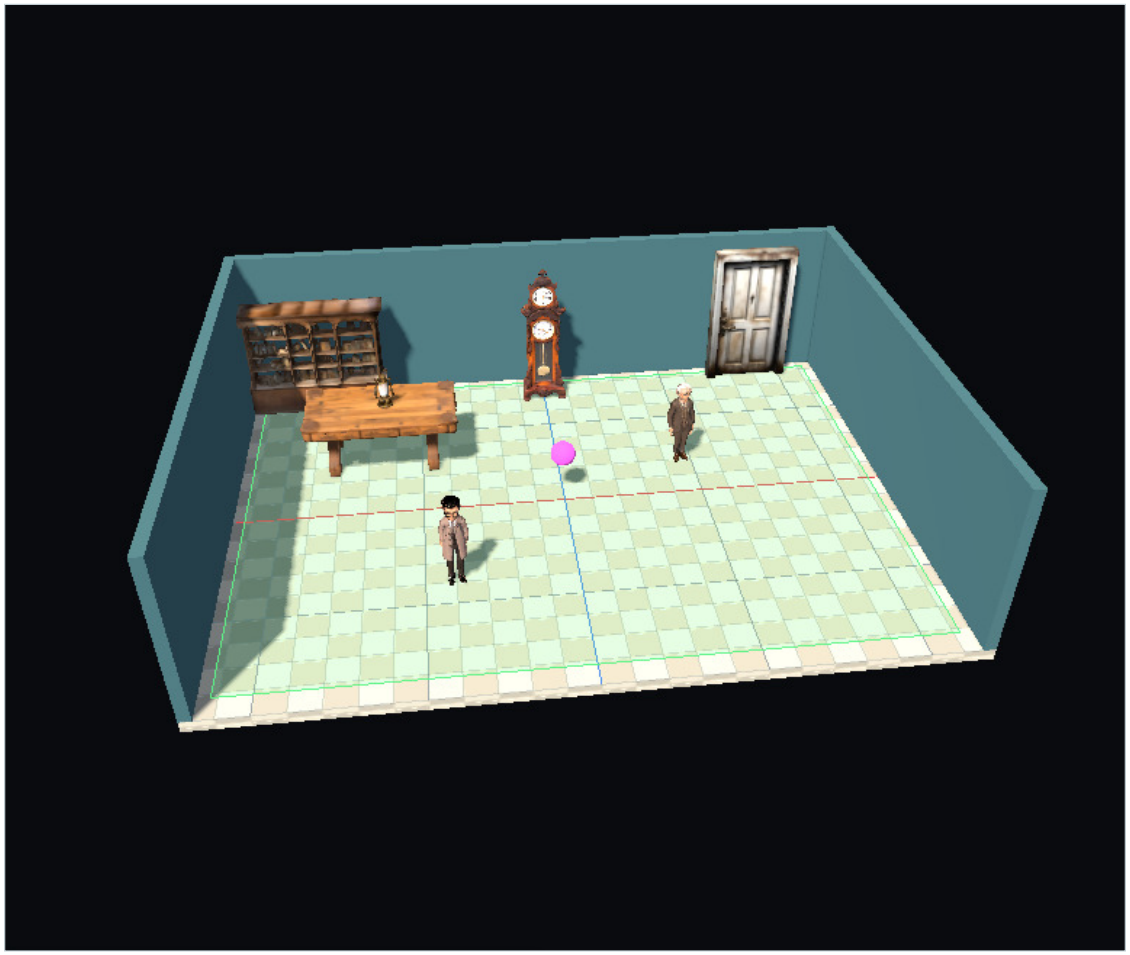

Figure 7. The same room shell with imported furniture. The navigation still needs to remain usable around the new obstacles.

6. Match the collider to the furniture

There are three separate questions: what the player sees, what stops the character, and what can receive a click. A large click target is sometimes helpful. A large physical obstacle can make the player stop much too far away.

For a solid clock, keep **Blocks Other Characters** on. In General, enable **Custom Box Collider** and fit it to the solid body of the clock.

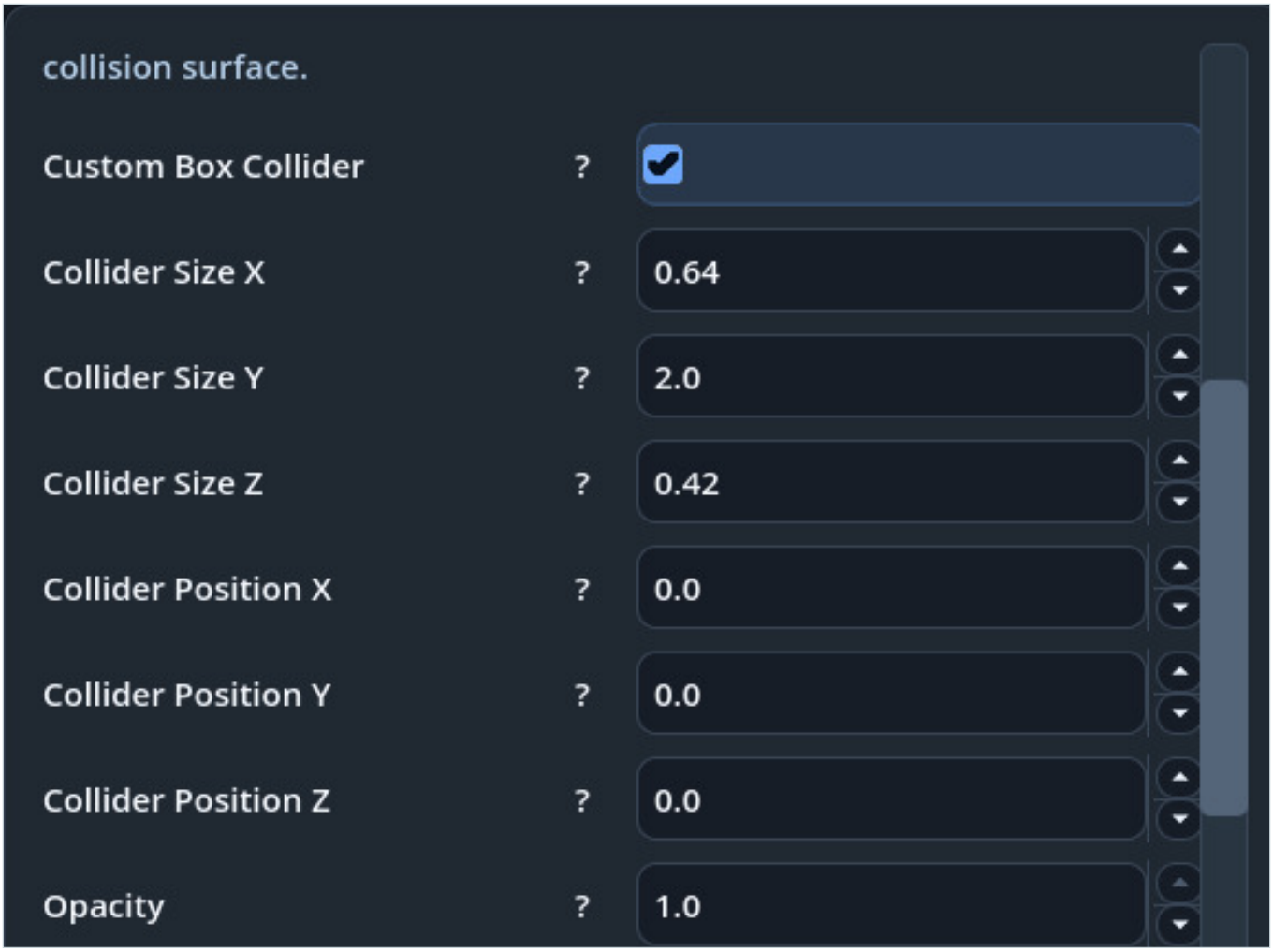

Figure 8. A custom physical box fitted to the clock. Its size and local offset are independent of the model file.

For this model, Collider Size **0.64, 2, 0.42** and Collider Position **0, 0, 0** are a close starting fit before the object's uniform scale is applied. The offset is relative to the object's origin, not a second world position.

For a table, a simple box around the whole tabletop can be a useful first approximation. It also blocks the space underneath. If you need characters to walk between individual legs, plan more detailed collision rather than expecting one box to do that job.

Do not use an event volume as a collider adjustment

Event Volume is the area used for entry, exit and related event tests. **Stop Distance** is part of interaction approach. Neither automatically makes a badly sized physical obstacle smaller.

Symptom	First thing to inspect
The visible clock is the wrong size	3D Scale.
The character stops well short of its body	Blocking collider size and offset, plus the character radius.

Symptom	First thing to inspect
Clicks are hard to land	The hotspot/click target and whether the object is interactive.
A collision-entry action fires too early	The Event Volume.
A Use approach stops too far away	Walk-to-Interact Behavior and Stop Distance, after checking collision.

Keep the decorative lantern's blocking off while it sits on the table. The table already provides the main obstacle there.

7. Import and inspect an animated character

An animated character is still imported as **Kind: Model**. Kind Animation is for image-animation assets; choosing it does not turn a skeletal model into a walking person.

1. Add a model asset in the Asset Designer.
2. Give it ID `player_model` and Name **Workshop Player Model**.
3. Browse for the animated model, let the import finish, then Apply Asset Changes.
4. In Asset Preview, open the **Animation** selector.
5. Choose an idle clip and press **Play**.
6. Choose a walk clip and play that too.
7. Check Loop and Speed in the preview. Use Speed 1 for ordinary playback.

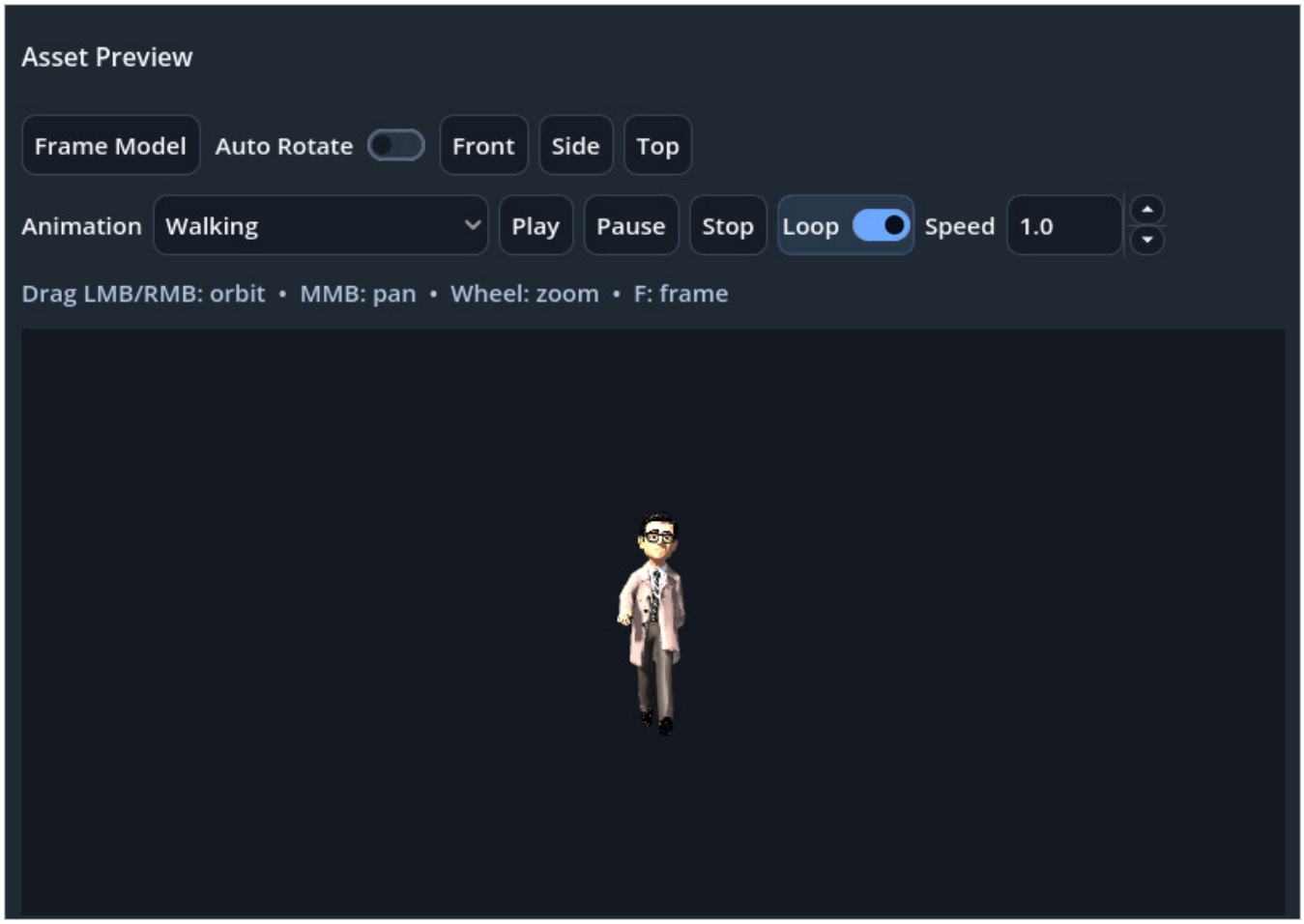

Figure 9. The imported character's real model clips are available in Asset Preview. Previewing a clip does not yet assign its gameplay job.

If no animation clips are detected, check the source export. A skeleton without animation tracks will not produce a walk cycle. Re-export with the required actions/clips included, or choose a file which already contains them.

Preview the clips rather than guessing from names. A clip might be called Walking, Armature|Walk, Take 001, or a long internal identifier. The name is whatever the model actually contains.

8. Give the player the animated model

Keep using the reusable Player character from the first guide. We are replacing its visible artwork, not adding another playable actor.

1. Open the **Character Designer** and choose **Player(player)**.
2. On the Character identity page, set **Model / Asset** to **Workshop Player Model(player_model)**.
3. Apply Changes and save.
4. In Character Settings, use Model Height about **1.8** and Collision Radius **0.35** for this example.
5. Keep Player's role and playable/default-starting settings as they were in the first guide.

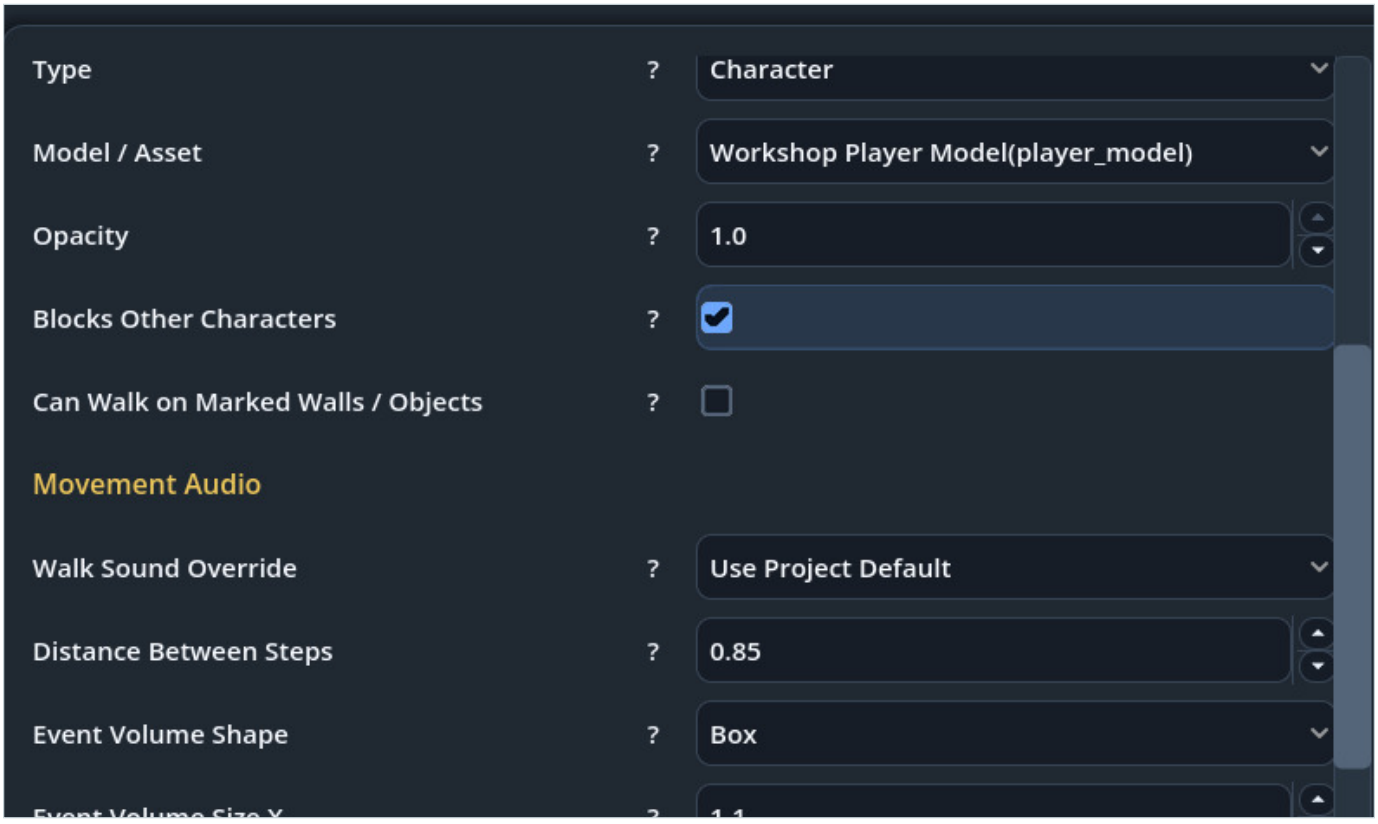

Figure 10. The reusable Player definition references an actual model asset.



Figure 11. Model Height and Collision Radius solve different problems. One fits the visible actor; the other affects its physical clearance.

Check an existing room instance too

Return to Scene Objects and select Player. In General, check that **Character** points to **Player(player)**. The model assignment itself is on the reusable definition; this scene-instance selector chooses which definition is placed here.

If an older placement still displays its previous artwork, note its position and save. Remove only that placement from Scene Objects, then use **Add to Scene > Character** to place Player again and restore Position **-2, 0, 2**, Rotation 0, 0, 0 and Scale 1, 1, 1. Keep the reusable Player definition in the Character Designer. This refreshes the placement from the updated character instead of deleting the character itself.

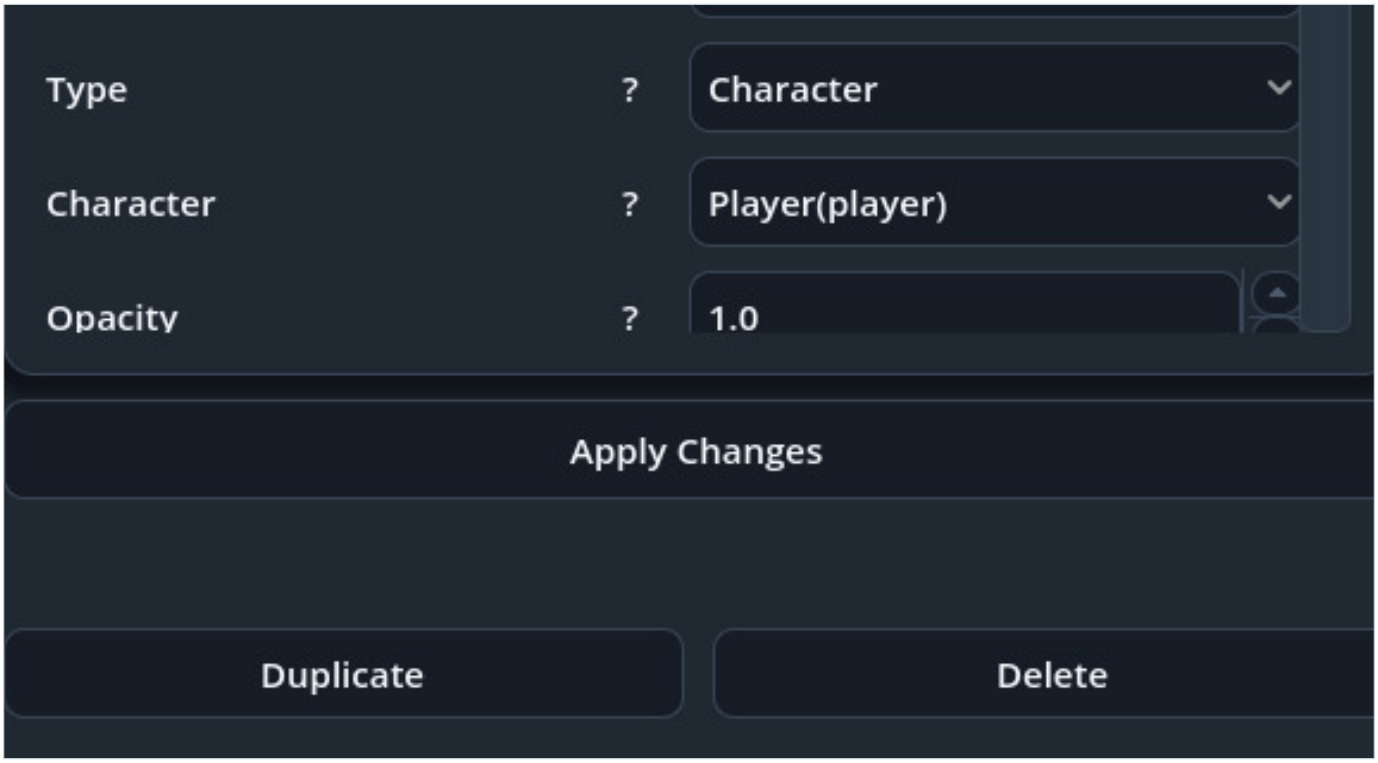

Figure 12. The room instance selects a reusable Character. Model artwork and animation states are edited on that definition.

For imported artwork, return Model Tint to **white** if you want its original material colours. The coloured placeholder tints from the first guide were only there to distinguish the simple capsules.

Do not use an ordinary Asset object as the replacement player. It may display the model, but it does not give it the reusable Character's walking, state or player-control behaviour.

9. Tell PCA 3D which clip means walk

The model contains clips. The Character uses **states**. A state such as `walk` tells the engine which imported clip to play for that gameplay job.

Open the player's **Animation States** section in the Character Designer. Select an existing state or click **Add State** for a missing one.

Set up Idle

1. Set State ID to `idle` and Display Name to **Quiet breathing** or **Idle**.
2. Choose **Model Clip** as Animation Source.
3. Choose the player's model in **Animation / Sprite Asset**, or keep **Current Character Model** when this state uses the character's ordinary assigned model. The pictures use that current-model option.

4. Choose a suitable idle clip in **Known Clip**. The Clip Name field should reflect the actual imported name.
5. Turn **Loop** on and set Playback Speed to **1**.
6. Click **Validate and Apply State**.
7. Set the character's **Default** state to Idle and save.

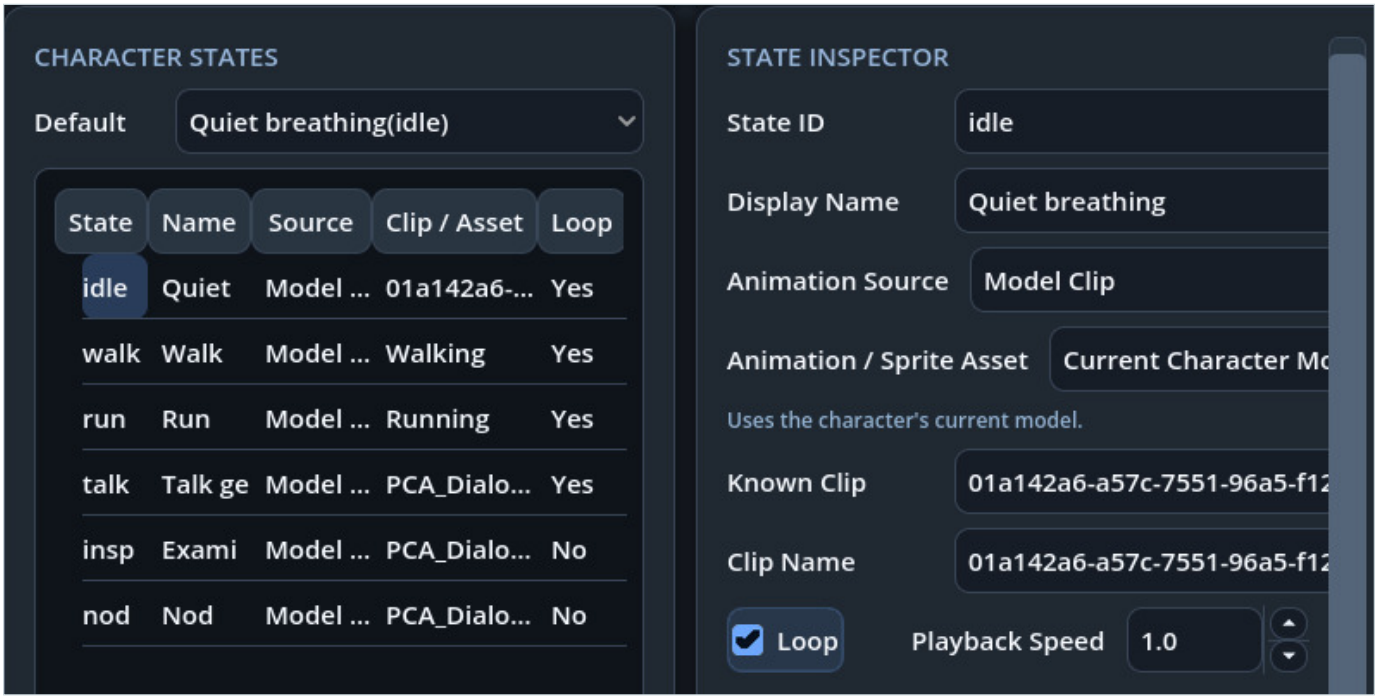

Figure 13. Idle maps a gameplay state to a real clip in the selected model. A long clip name is not an error.

Add Walk, Run and Talk

Use the same process for each state. The state IDs below are the gameplay names; the Clip column describes what to choose from your particular model.

State ID	Imported clip to choose	Loop	Speed
idle	Calm standing/breathing animation	On	1
walk	The model's walking cycle	On	1
run	Its running cycle, if supplied	On	1
talk	A speaking/gesture clip	On for this example	1

State ID	Imported clip to choose	Loop	Speed
inspect	A short examine/consider gesture	Off	1
nod	A short nod	Off	1

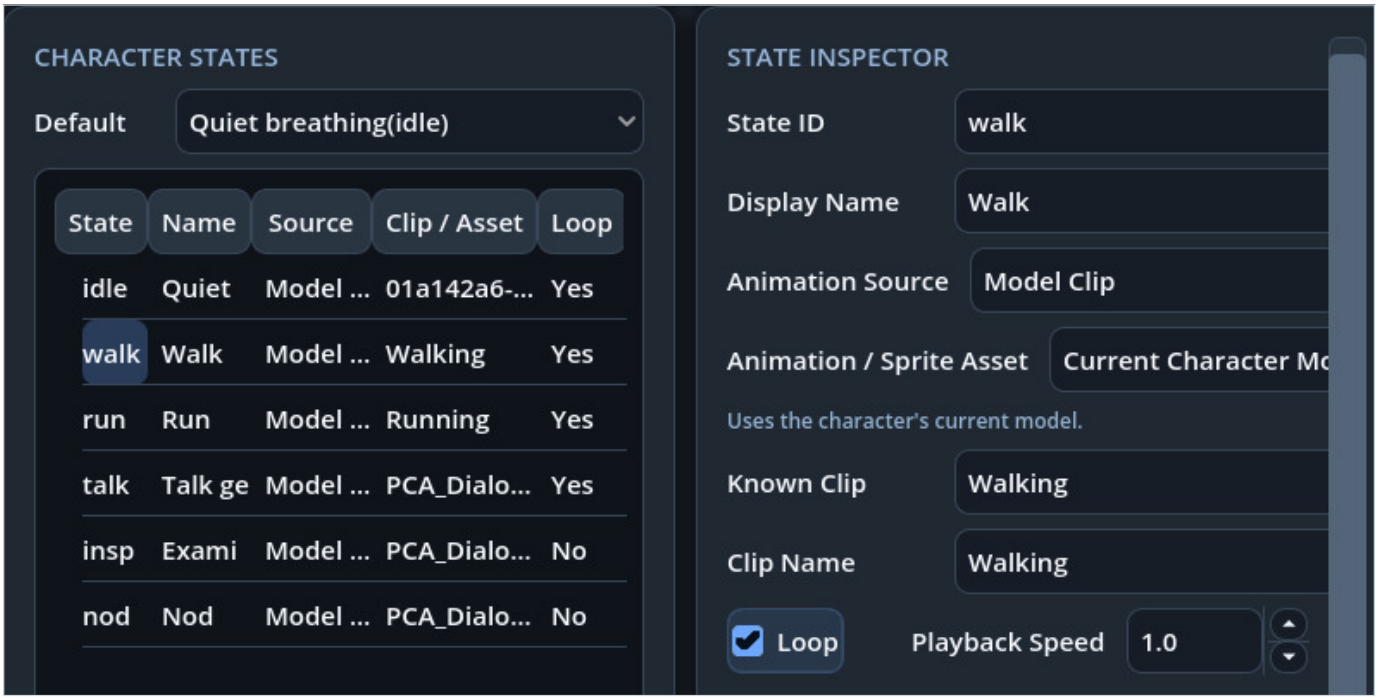

Figure 14. The walk state uses the imported Walking clip. Setting the display name alone would not create this link.

The supplied player model's walk and run clips are **Walking** and **Running**. Its quiet-breathing clip has the internal name `01a142a6-a57c-7551-96a5-f1229d38cb51`. Its inspect gesture is **PCA_DialogueConsider**, and its short nod is **PCA_DialogueNod**. Your model does not need to use these names.

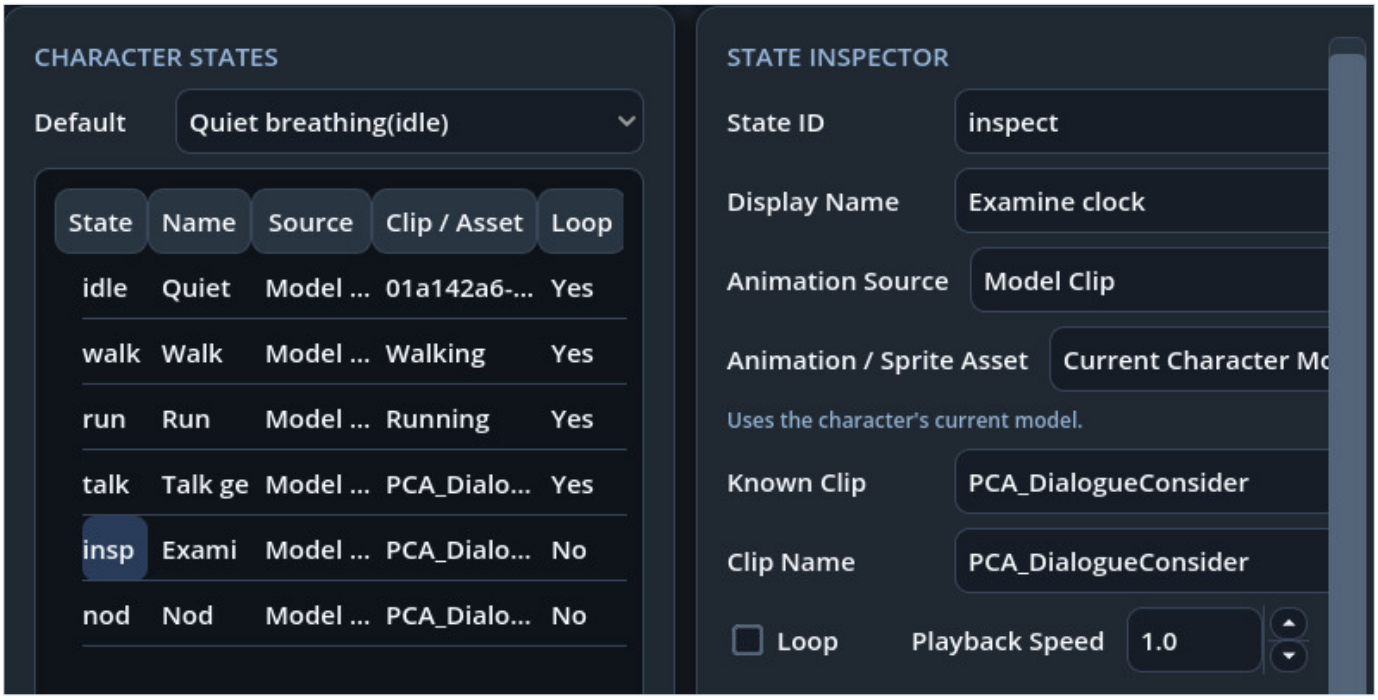

Figure 15. A one-off gesture has Loop off. It can later be started by a Play Animation action using the state ID inspect.

Apply each state before selecting another. Use the fields for ordinary changes; you do not need to edit State JSON. **Preview State** can restart the runtime scene to test the selection, so apply and save useful work before using it.

If the actor slides or faces the wrong way

Select the placed character in Scene Objects, open **Visual Adjustments**, and use **Lock Root Motion** when the walk clip moves its visual root and would otherwise drift away from the navigation body. Navigation moves the actor; the clip should not accidentally move it a second time.

Model Facing Offset corrects the model's forward direction. It is a model-facing correction, distinct from the placed object's ordinary 3D rotation. Check movement from several angles instead of rotating the whole room to compensate.

Walk Speed controls travel through the room. Playback Speed controls how quickly the animation runs. Adjust them separately if the feet are skating. Changing Collision Radius is not a fix for a fast walk cycle.

10. Upgrade the NPC and test movement

Use the same Character Designer workflow for Mara. Assign an animated NPC model, map its own clips, and keep its Role as **Non-Player Character**. Leave Playable Character and Default Starting Character off.

When copying a state setup, check **Animation / Sprite Asset** in every copied state. A state accidentally referencing the player's model can use the wrong source artwork. Choose the keeper's own asset and clips.

For the pictures, Mara stands at **2.3, 0, -1**, the player begins at **-2, 0, 2**, and both have Model Height 1.8. Their scene Scale stays 1, 1, 1. Their source models have their own original materials.

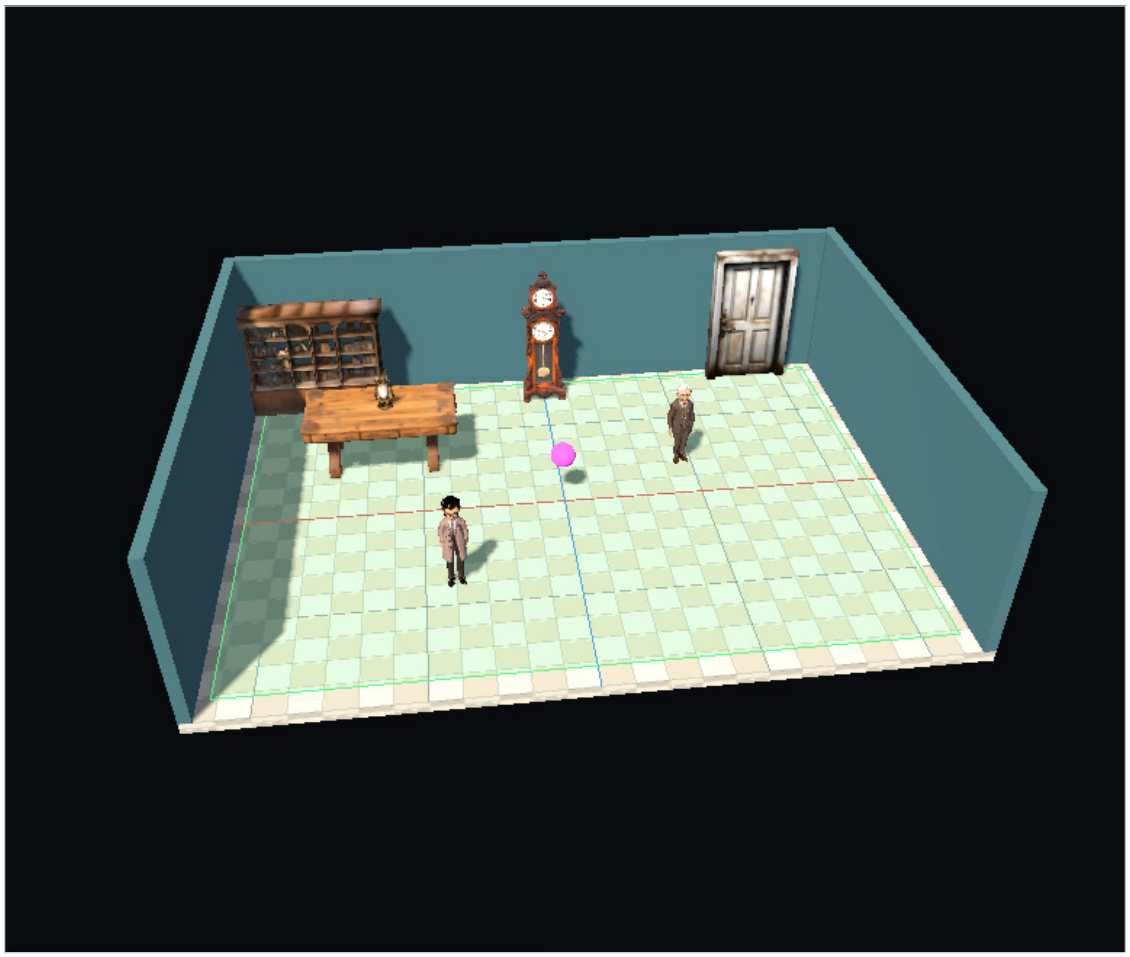

Figure 16. The two character placements now use real model artwork instead of capsules.

Test before adding a puzzle

1. Apply the character and state changes, then Save Project.
2. Select the workshop and click **Test Scene**.
3. Check that both models are visible and their feet meet the floor.
4. Click several reachable floor positions, including a route around the table.
5. Watch the player change from idle to walking and back to idle.
6. Use Stop before returning to editing.



Figure 17. The room running with the imported furniture and animated character models.

A visible model with no walking animation usually points to state mapping, not to the camera. A complete absence of the model points first to the model reference, visibility, scale, position or a scene-instance override.

> **Before you continue**
> Finish this movement check before adding a long interaction. It is much easier to diagnose one missing walk clip than a missing walk clip mixed with a camera change, trigger and puzzle condition.

11. Give the clock a Look response

A model is scenery until you attach behaviour to it. We will start with one action and then build a sequence.

1. Select **Workshop Clock(clock)** in Scene Objects.
2. Open **Verb Actions**.
3. Find the **Look Verb** group. Do not put the line in Walk or Talk by mistake.
4. Choose **Show Message** for its first action.
5. Enter: **The pendulum has stopped. A small wheel has slipped out of place.**
6. Set Duration to about **3.5 seconds** and turn Block Player Input on for this observation.
7. Apply and save.

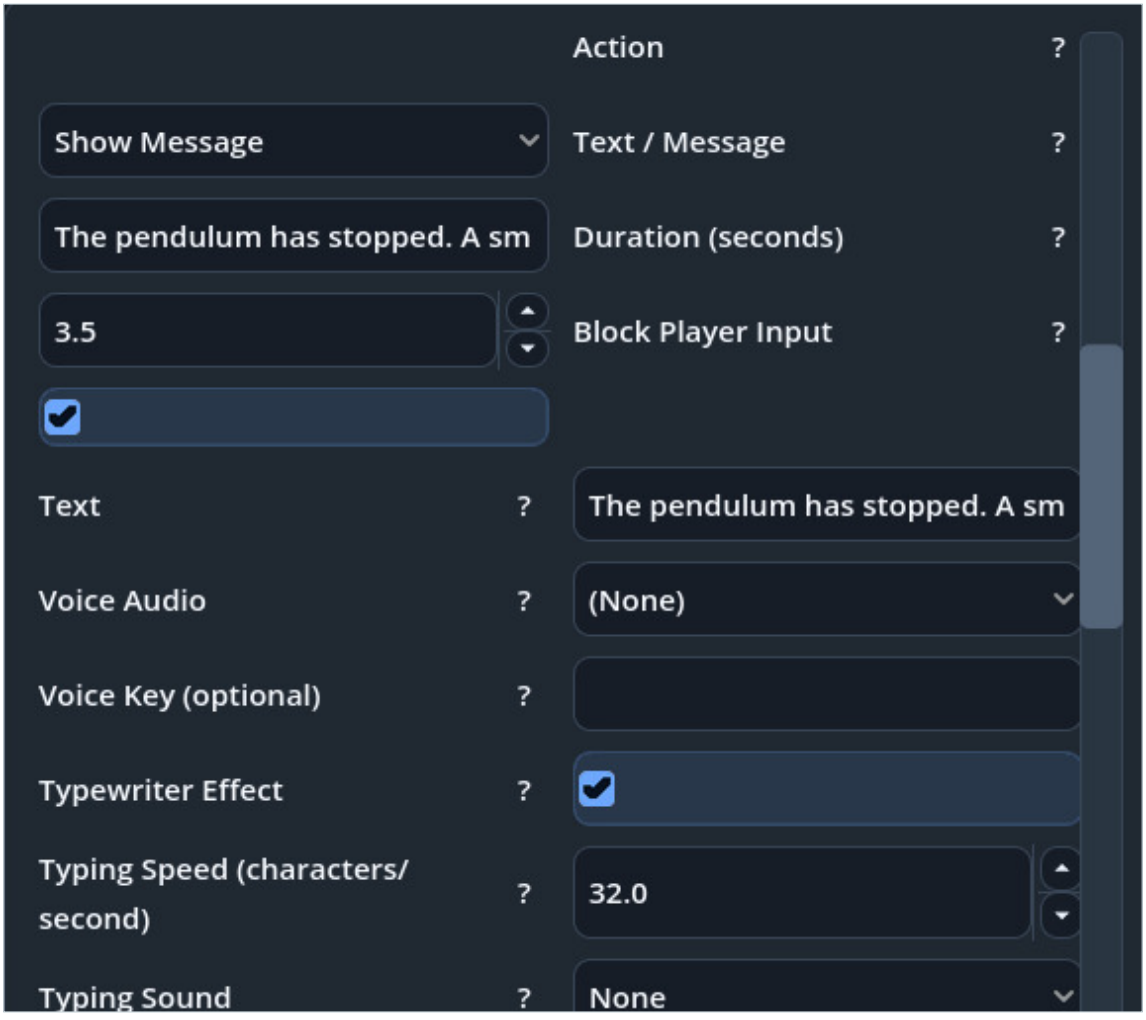

Figure 18. The Look action belongs to the placed clock. The first response is deliberately simple.

For an optional typewriter reveal, turn **Typewriter Effect** on and use a Typing Speed around **32 characters/second**. Choose **none** for Typing Sound in this quiet example. Default uses the built-in blip; custom lets you choose an audio asset.

In the first guide's context-sensitive interaction mode, **Default Verb** determines what an ordinary click runs for this object. Set it to Look while testing this response. Later we will change it to Use. In a verb-selection mode, select Look in the game UI before clicking the clock.

Pointer events can take priority

The Object Events tab also has Left Click and other pointer-event lists. An authored pointer event can take priority over ordinary verb interaction. For this lesson, leave the clock's pointer-event lists empty and put its behaviour in Verb Actions.

If you intentionally need a pointer event to forward to the selected verb, use **Execute Selected Verb Command** where that event surface offers it. Do not accidentally create two different click systems and expect both to run.

12. Add a close-up camera

Keep Room View as the room's Default Camera. Add a second authored camera for the repair sequence.

1. In Add to Scene, choose **Camera**.
2. Select the new camera under Cameras in Scene Objects.
3. Give it ID `clock_close` and Name **Clock Close-up**.
4. Keep Perspective and Enabled at Runtime on.
5. In Position & View, enter the values below.
6. Apply and save.

Setting	X	Y	Z
Position	0	3	1.4
Look Target	0	1.5	-3.7

Use Field of View **40**, Near Clip about **0.05**, and Far Clip **100** for the pictured room.



Figure 19. Clock Close-up aims at the clock from inside the room. It is an extra shot, not the new default gameplay camera.

Select the camera and check its preview. A different clock size or position needs a different shot. If a wall or furniture fills the picture, move the camera or its target deliberately.

Change Camera selects another shot immediately. **Activate / Blend to Camera** can move between authored views over a chosen duration. We will use the latter so the close-up has a short transition.

13. Create a variable for the repair

A variable is a named piece of game memory. Here we need one yes/no value: has the clock been repaired?

1. Open the **Variables** designer.
2. Add a variable with ID `clock_fixed` and Name **Clock Repaired**.
3. Choose **Boolean** as its type.
4. Leave its starting value **false**.
5. Apply Changes and save.

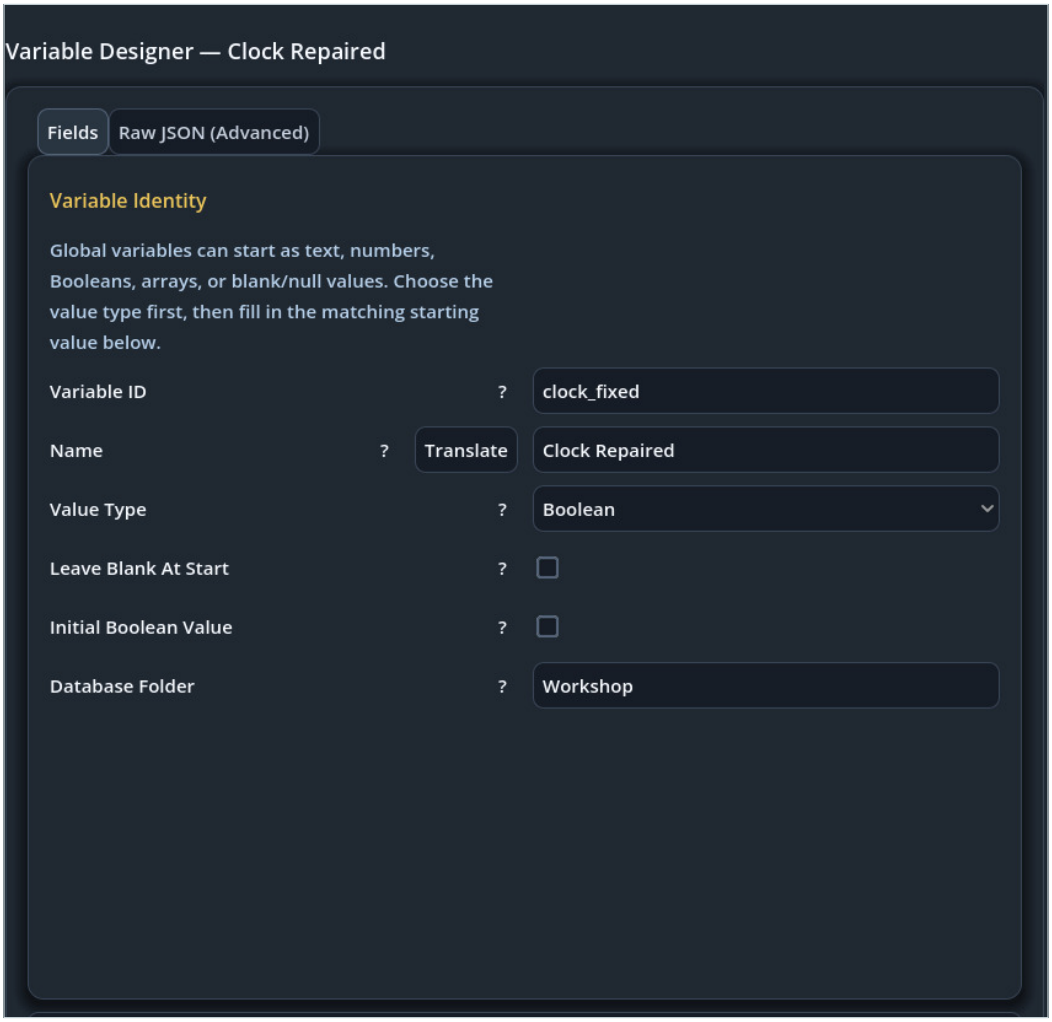

Figure 20. The project starts with Clock Repaired false. Runtime actions will change the current game value.

Boolean false means no; true means yes. It is different from the text string "true". When an action asks for a generic value, enter the actual boolean value rather than quoted text.

The designer holds the starting value for a new run. An action changes the current game's value. Ordinary game saving can preserve that current state; testing from a fresh start uses the project's initial value again.

14. Build the clock's Use action list

An action list runs its steps from top to bottom. You do not need a visual graph just to perform a straight sequence.

Select the clock and open Verb Actions. Change Default Verb to **Use** for the context-sensitive version of this lesson. Find the **Use Verb** group and set up these actions in order.

Step	Action	Important fields
1	Activate / Blend to Camera	Camera: Clock Close-up; Blend Seconds: 0.7; Wait for Blend: on.
2	Play Animation	Room: workshop; Character / Object: Player; Animation: inspect.
3	Show Message	One small adjustment should do it.; Duration 2.5; Block Player Input on.
4	Play Sound	Sound Asset: your chime; Channel / ID: clock_chime.
5	Change Variable	Variable: Clock Repaired; Operation: set; Value: true.
6	Activate / Blend to Camera	Camera: Room View; Blend Seconds: 0.7; Wait for Blend: on.
7	Show Message	The clock is running again.; Duration 2.5; Block Player Input off.

Import a suitable audio file as an Audio asset for Step 4. The pictures use the bundled Chime with asset ID `clock_chime`. A channel name is a runtime label for controlling that sound later; it is not the sound asset itself.

Add the steps in the correct group

1. Set the Use group's first Action selector to Activate / Blend to Camera.
2. Fill its camera, duration and wait fields.
3. Use **Add Next Action** at the end of the Use group to append Step 2.
4. Choose Play Animation and fill its target and animation.
5. Continue appending the remaining steps in the order above.

6. Apply Changes and Save Project.

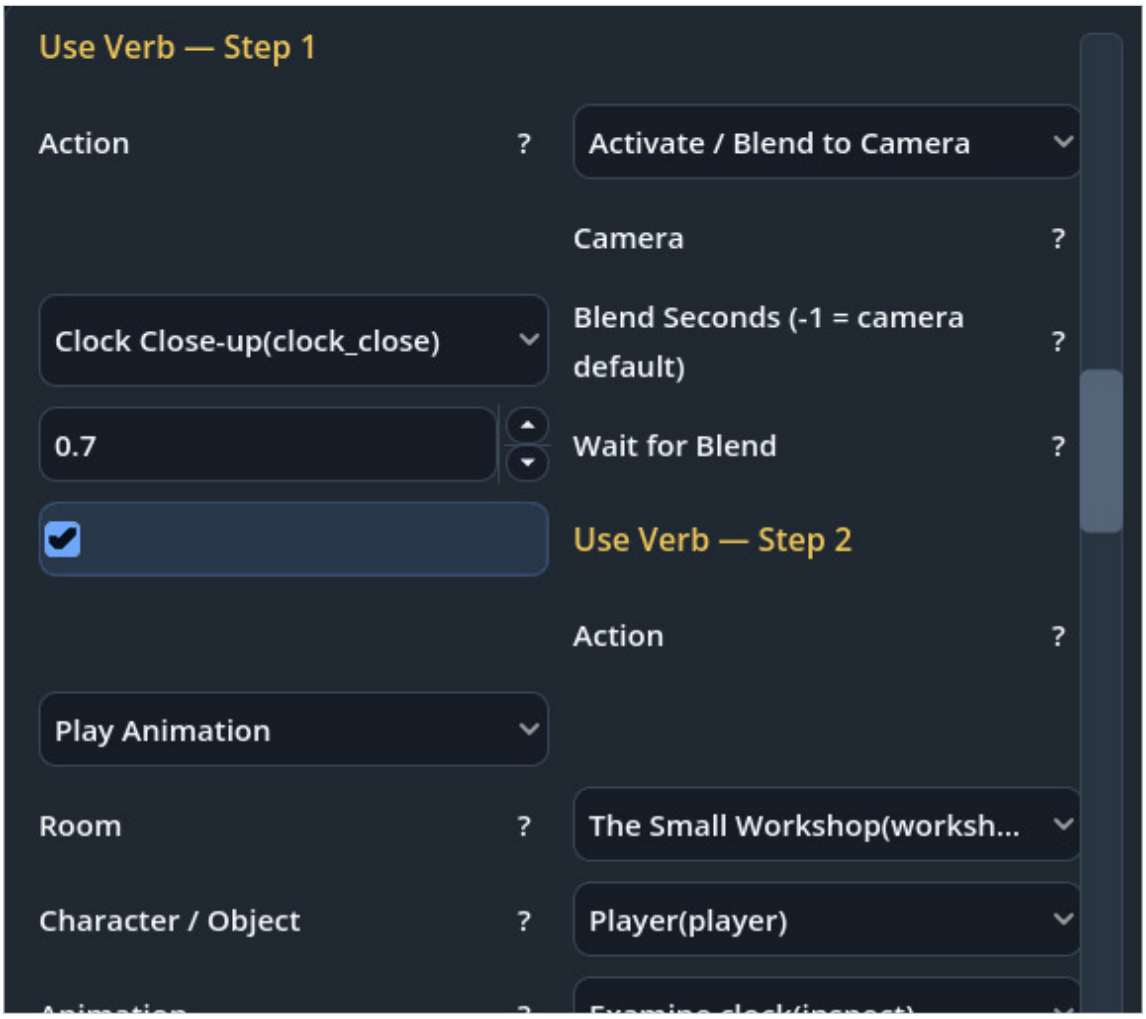

Figure 21. Step 1 enters the closer shot. Wait for Blend lets it finish before the next action starts.

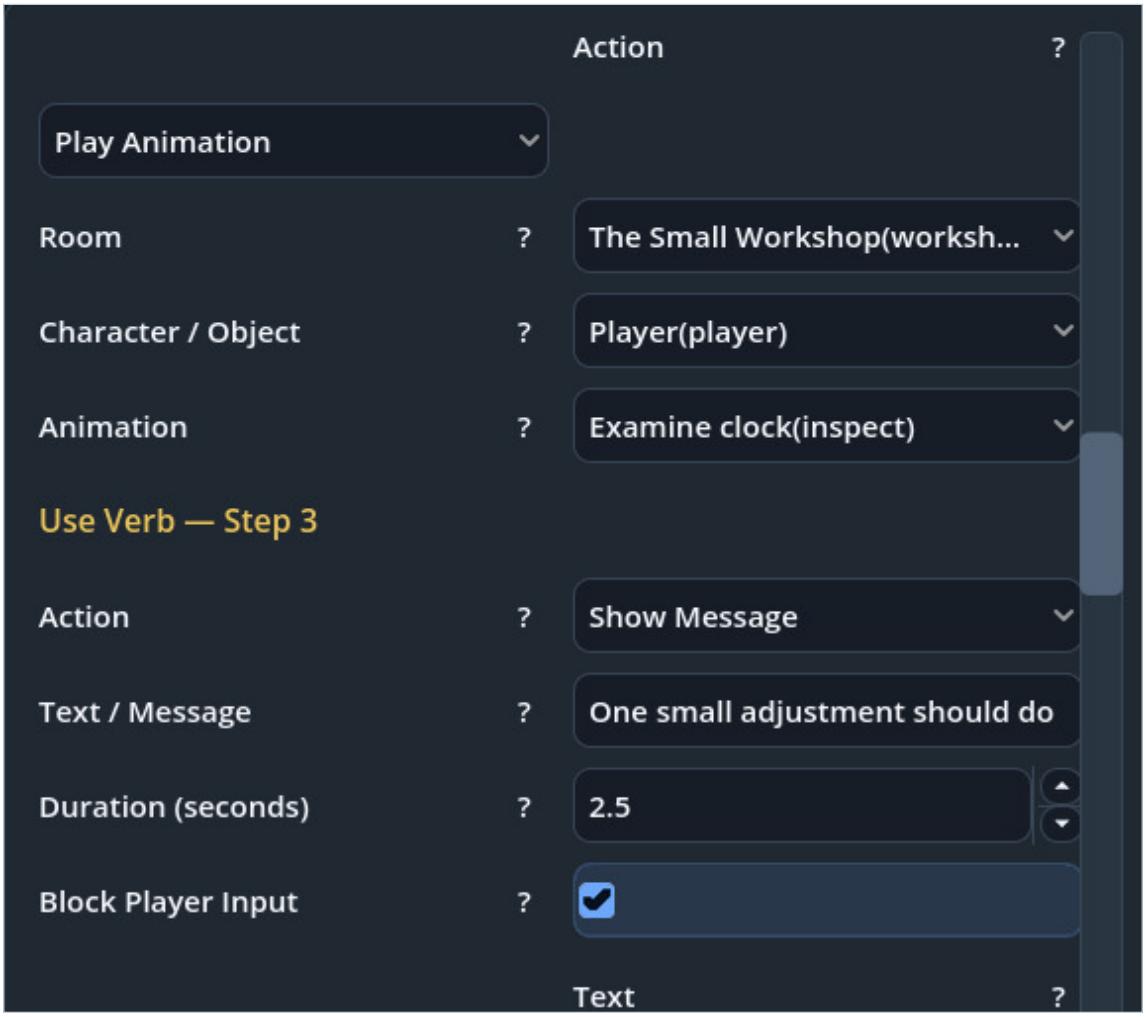

Figure 22. The animation target is the Player character. Animation uses the authored state ID inspect.

The inspect state starts the imported gesture. Do not assume that Play Animation waits until every frame of a clip has finished. Here the following timed message keeps the close-up on screen while the gesture plays. A visual script can use a Wait node when you need an explicit timed pause.

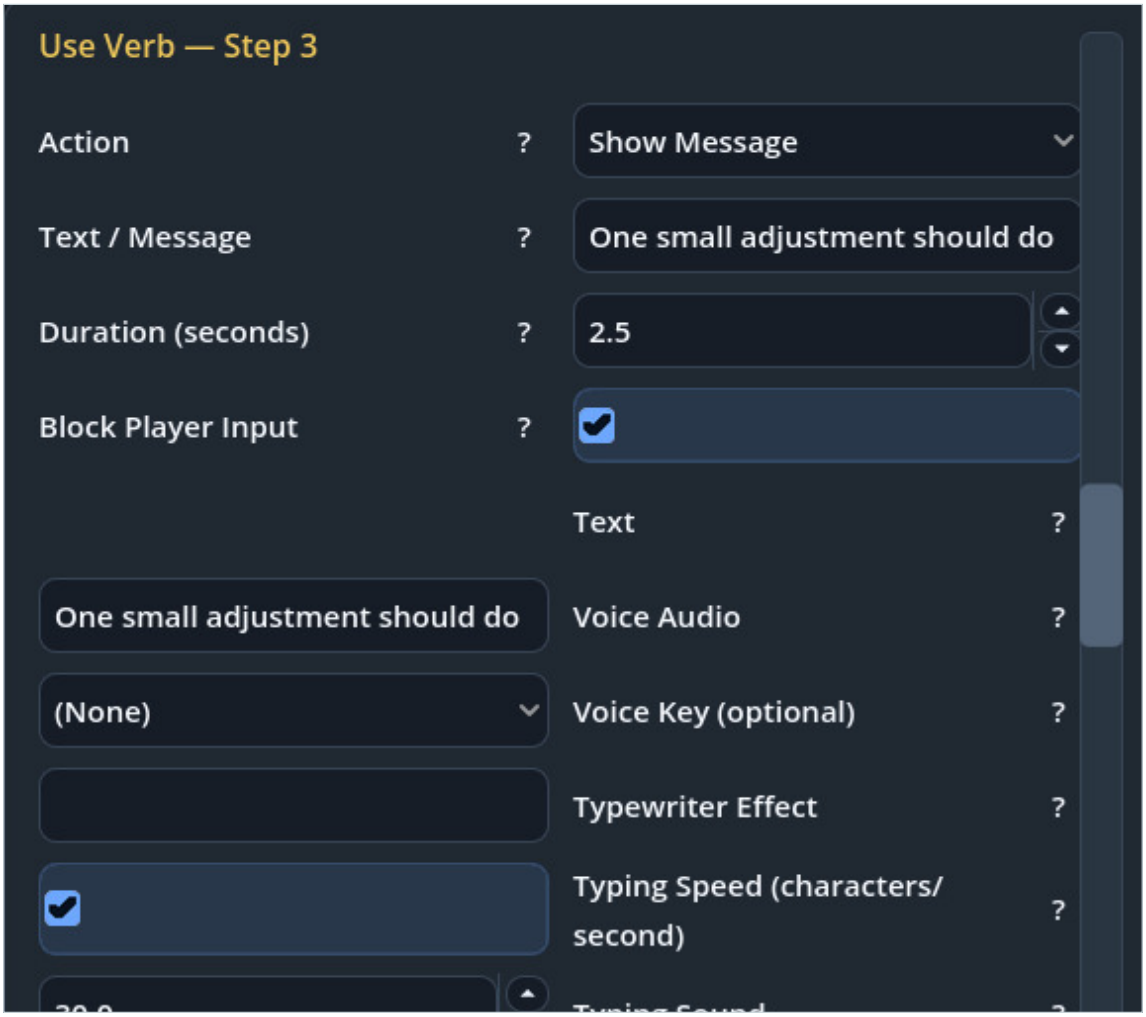

Figure 23. The timed observation holds the moment while the gesture plays. It is a message, rather than a new dialogue tree.

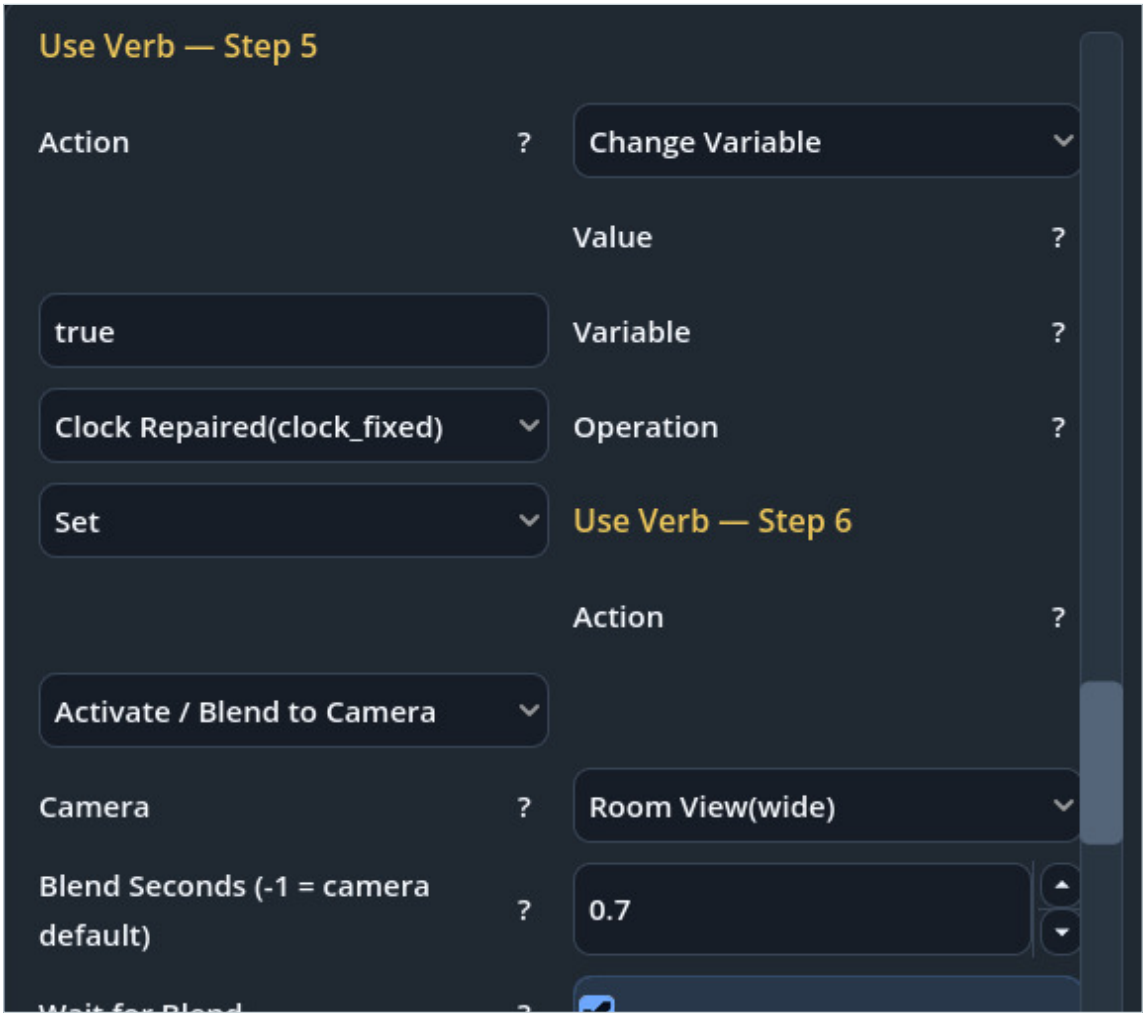

Figure 24. The repair changes the game's boolean value to true. Choose Operation set for an explicit result.

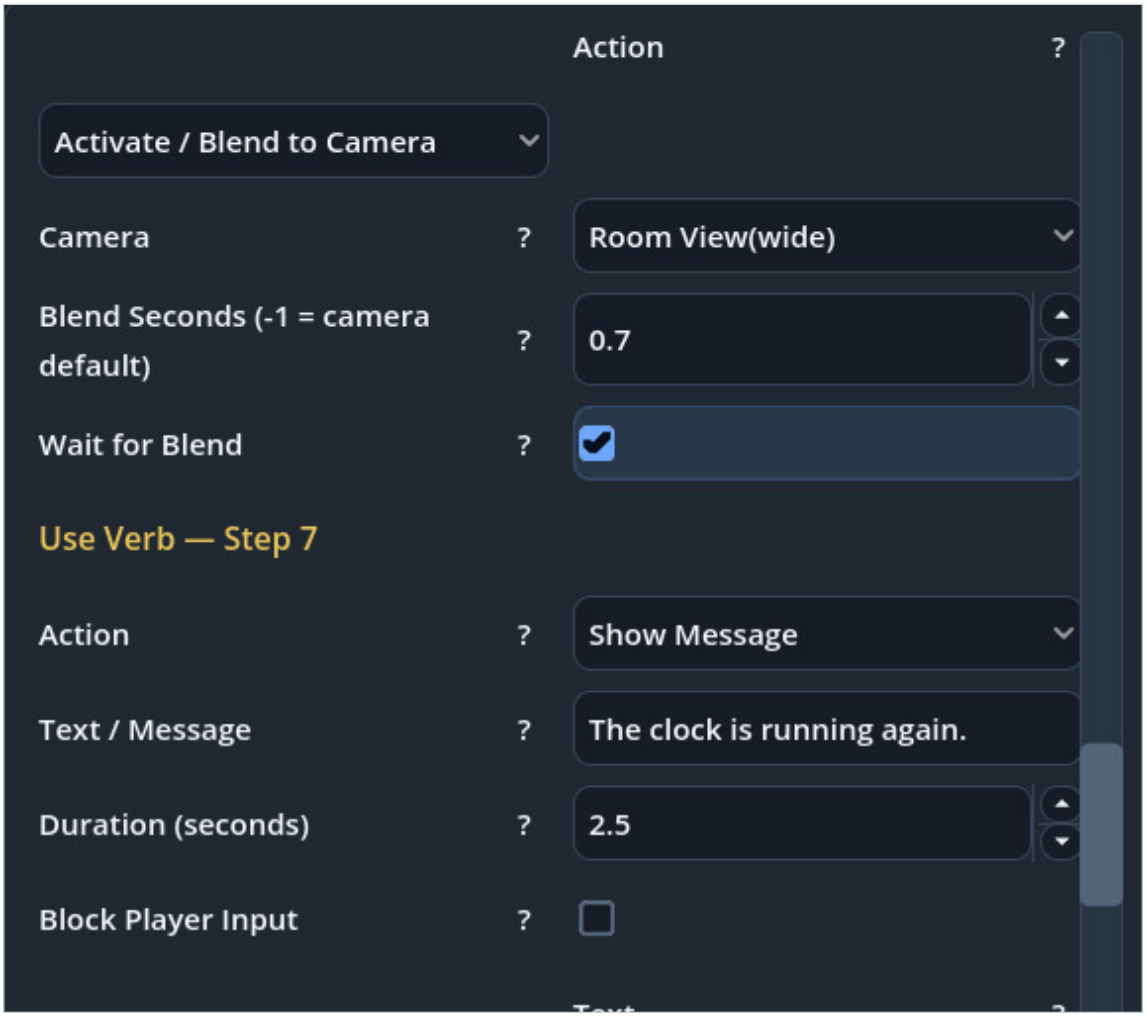

Figure 25. The later camera action restores Room View. Choosing a close-up does not automatically schedule this return for you.

In this Inspector, choosing **No Action** and applying the record removes that step's authored action. Add Next Action appends another one. Build a short sequence first and test it before extending it. Do not assume that every sequence widget in the editor has identical reordering controls.

> **Before you continue**
> Confirm the final camera action is present. A sequence which enters a close-up but never restores a useful gameplay view can feel broken even when every other step worked.

15. Let the character approach the clock

A click should not make the player repair the clock from across the room. Select the clock's General tab and find **Interaction Approach Override**.

Set **Walk-to-Interact Behavior** to **Always Walk to This Object**, set Stop Distance to about **1**, and leave **Face This Object after Approaching** on. Apply and save.

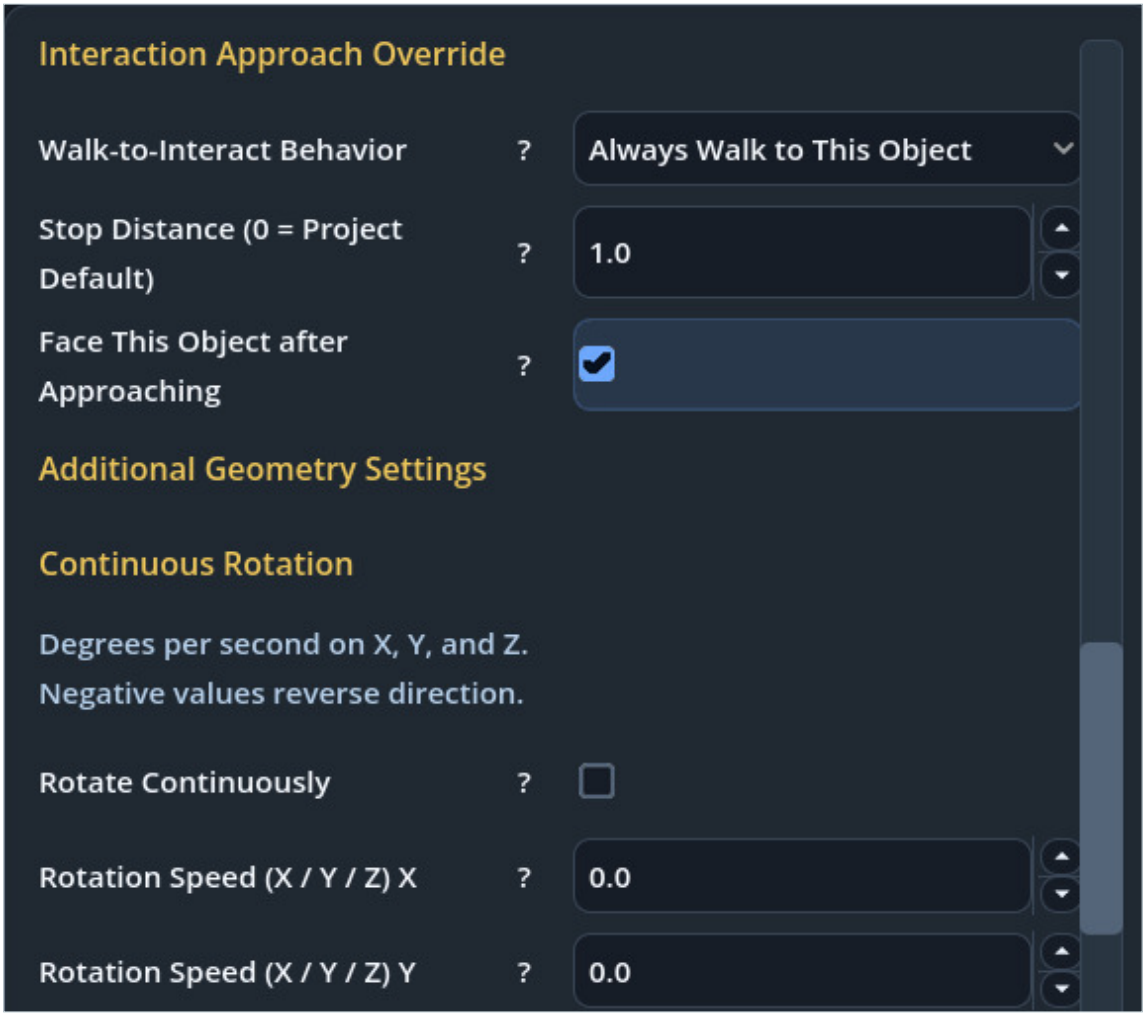

Figure 26. The object can override the project's approach behaviour. Stop Distance works alongside the physical collider, not in place of it.

These values work with the pictured clock and a character radius around 0.35. A different model or collider needs a different comfortable distance. The player should stand near the front of the clock, not inside its body.

A correct approach needs an accessible route. The clock must be next to the walkable area, and its own collider must leave somewhere for the character to stand. Reducing Stop Distance cannot force the character through a wall or an oversized obstacle.

If you want an exact authored approach point in a more elaborate scene, plan that position and route explicitly with movement actions. The simple automatic approach is enough for this small workshop.

16. Make the NPC speak

Mara can now use her imported model's Talk state while speaking. A full branching conversation is not required for a short response.

1. Select Mara's room instance.
2. Open Verb Actions and keep Default Verb set to Talk for a context-sensitive click.
3. In Talk Verb, choose **Character Say**.
4. Set Speaking Character / Speaker to **Mara(mara)**.
5. Enter: **Try the clock. Then check the archive door.**
6. Leave Wait for Continue off for the timed response used here.
7. Optionally turn Typewriter Effect on and set Typing Sound to none.
8. Apply and save.

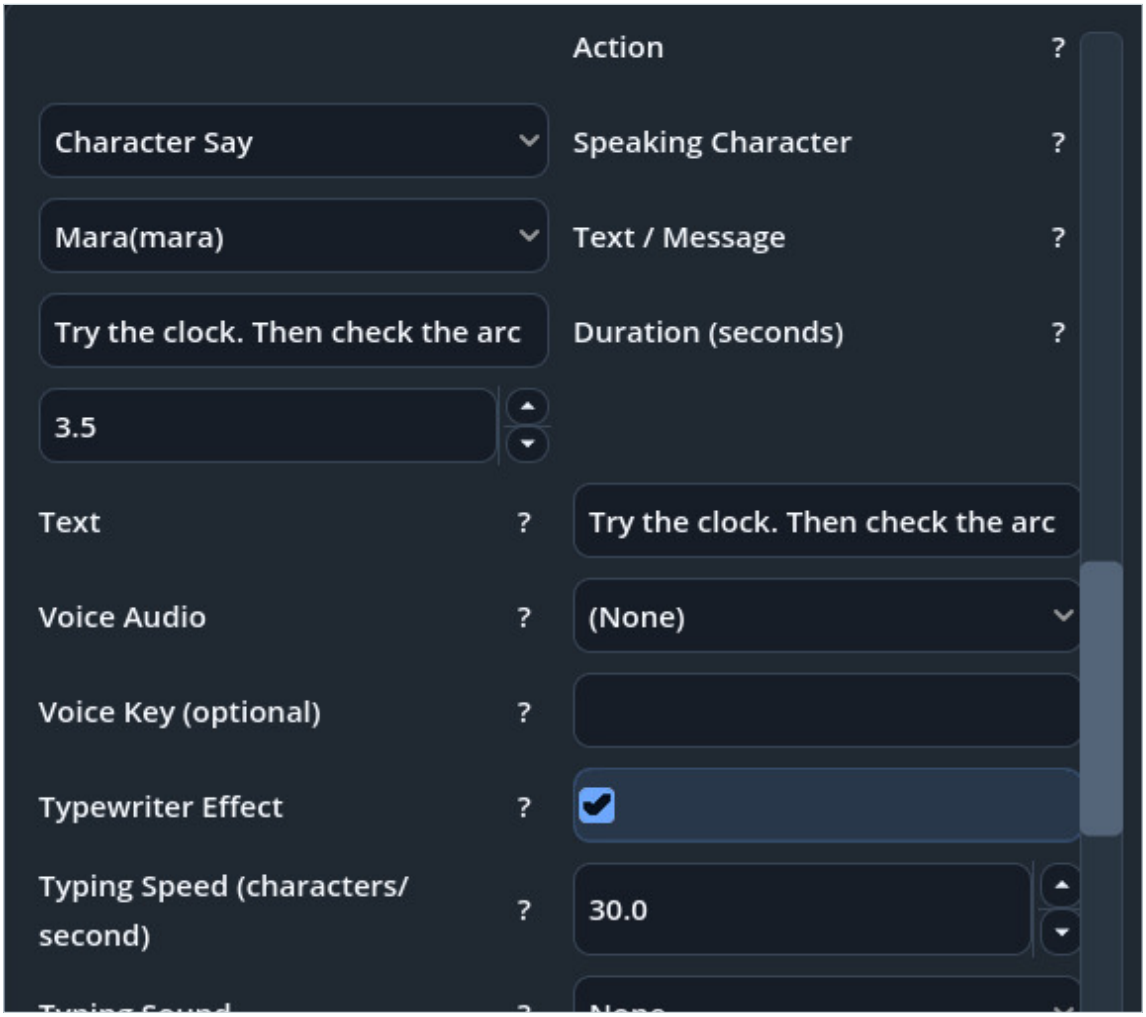

Figure 27. Character Say identifies a speaker. A generic Show Message is useful for narration, but does not replace this character-speaking setup.

The character's talk state should point to its own speaking clip. A silent text response can still animate the character. Recorded voice and lip sync are additional choices; a text line does not automatically create a voice recording or viseme track.

If you choose Wait for Continue, test how the player advances that line in the current GUI. Leave it off for an automatically advancing observation while you are first checking the scene.

17. Test the complete repair sequence

Save, open the workshop and use Test Scene. First walk around the imported props. Then use the clock.



Figure 28. The furnished room before the repair. Test the complete sequence in the running game, not only in camera preview.

Check the order: the player approaches, the camera blends closer, the gesture starts, the observation appears, the chime plays, the flag changes, and the camera returns.



Figure 29. The repair uses the closer authored camera while the character performs the observation.

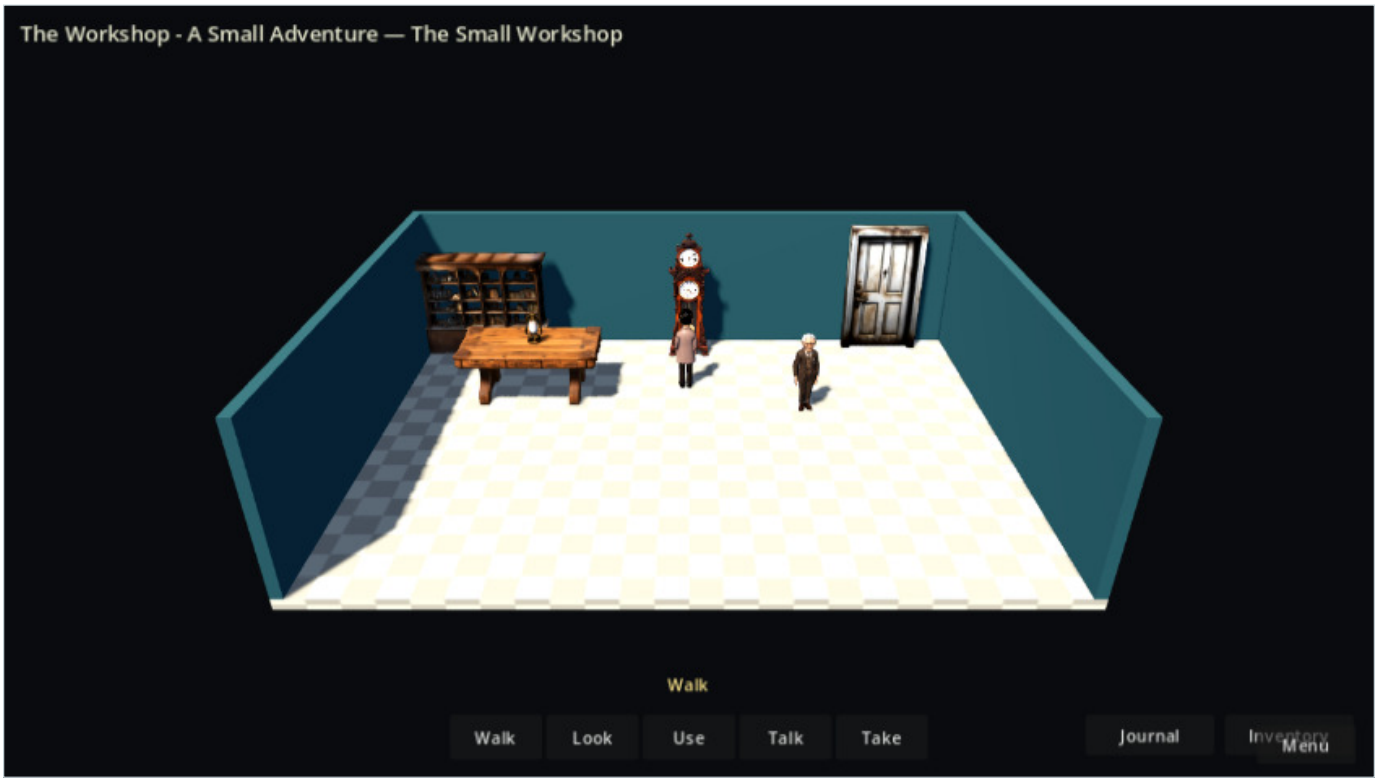

Figure 30. The room view returns after the repair. The repair flag is now true in this running game.

Talk to Mara as well. Check the visible model, speaking text, timing and return to normal play.

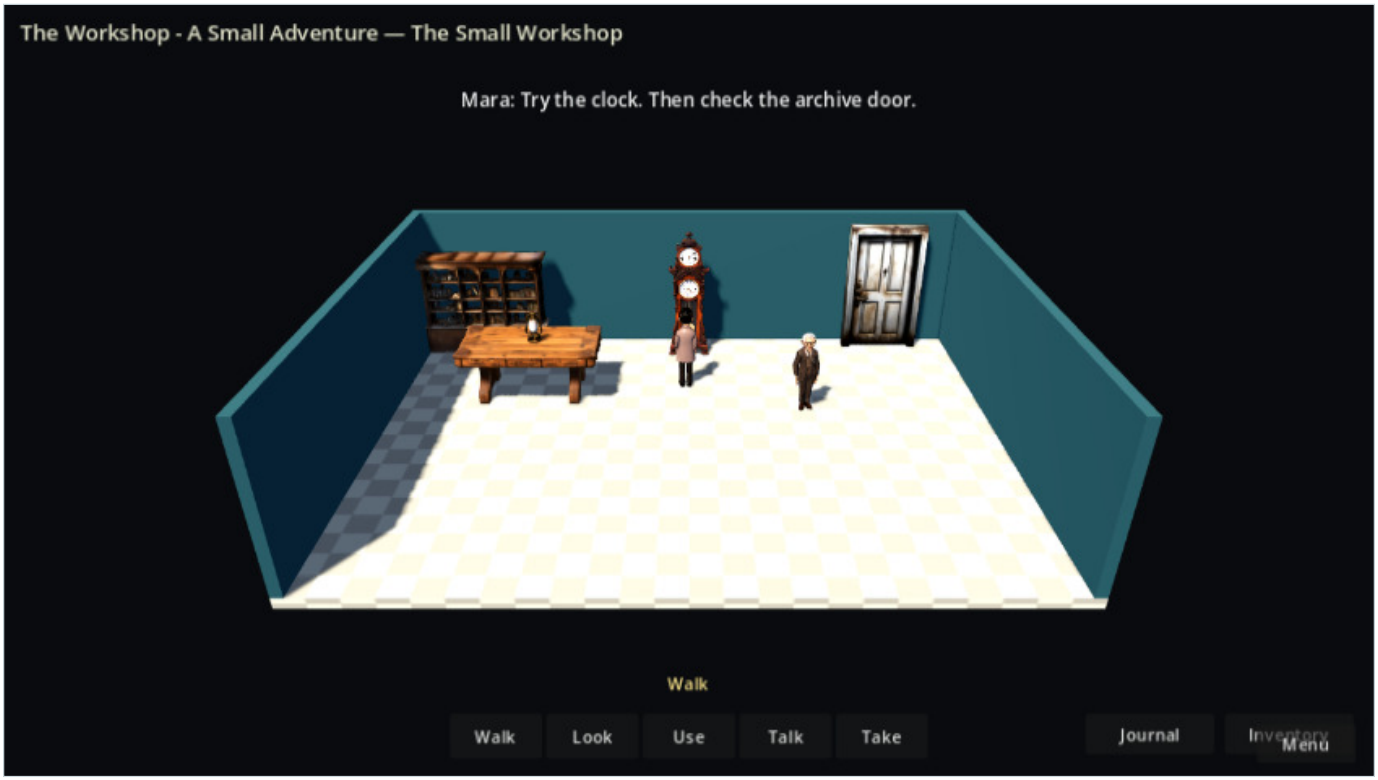

Figure 31. Mara's short speaking response in the actual game view.

This first straight action list can run again if the clock is used again. A true flag does not automatically disable an action list. The next chapter uses that flag to make another object behave differently.

18. Make the door check the repair flag

We will give the archive door two responses. Before the repair it refuses entry. After the repair it removes the door prop and reports that it is unlocked. This is a small visual-script example with a real yes/no branch.

It is a prototype opening effect: the prop disappears. It does not model a hinged door animation, cut a hole in the back wall or create a second room. Those need their own art, geometry or scene-change setup.

Build Check the Archive Door

1. Open **Visual Scripts** and add a script.
2. Give it ID `check_door` and Name **Check the Archive Door**.
3. Add an **On Interact** node and set its Verb to Use.
4. Add **If Variable**. Choose Clock Repaired and set Equals to true.
5. Connect the start node's next output to If Variable.
6. Connect the **false** output to a Show Message node: **Repair the clock before opening the archive.**
7. Connect the **true** output to **Enable / Disable Object**. Choose the workshop's Archive Door and turn Enabled off.
8. Connect that node's next output to another Show Message: **The archive door is unlocked.**
9. Use Save Project after finishing the graph.

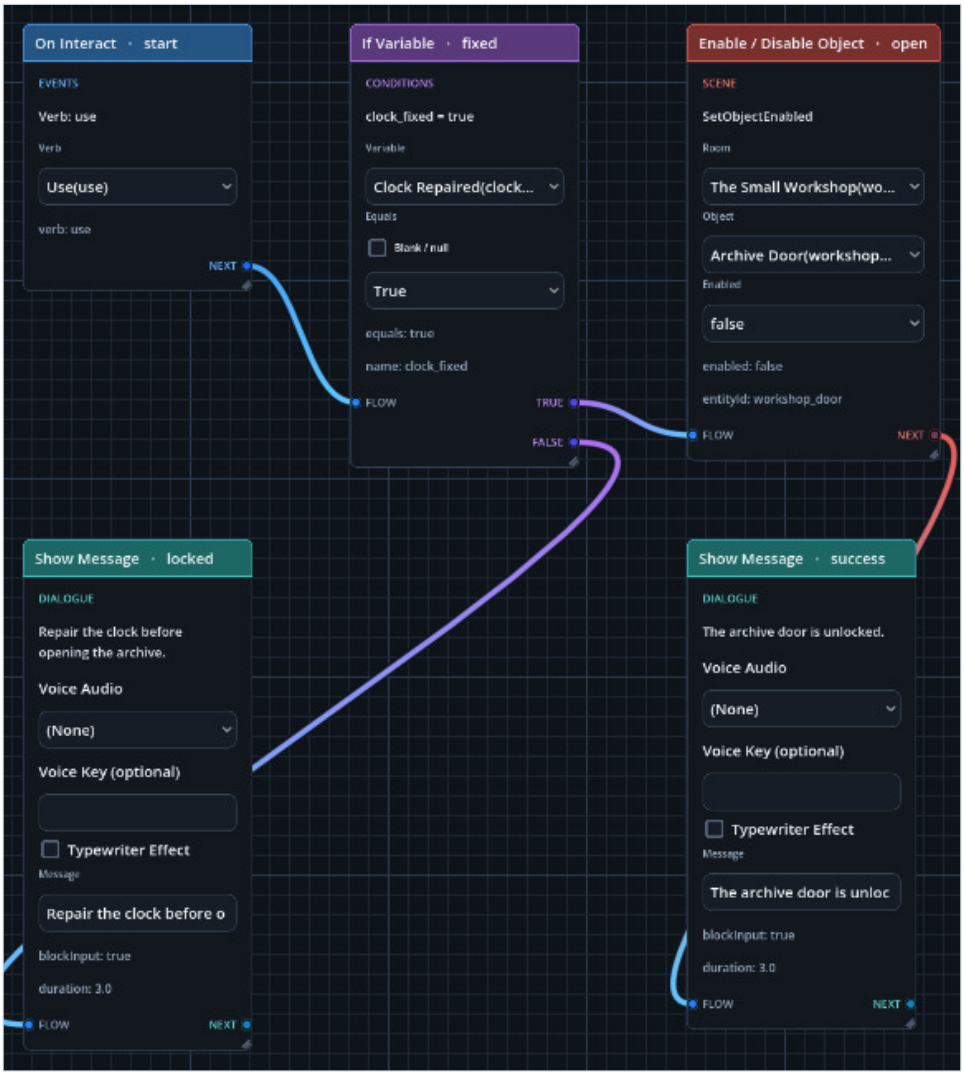

Figure 32. A condition chooses one route. The false route gives the refusal; the true route disables the door prop and shows the success message.

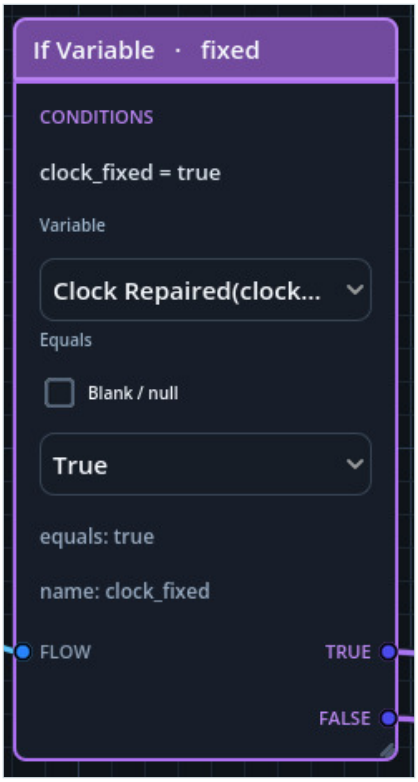

Figure 33. If Variable compares the current boolean game value with true. It does not set the value.

A node with no outgoing connection ends that route. Keep the two responses on their separate outputs. Connecting both after a normal next output would be a sequence, not a choice.

Call the graph from the door

1. Return to the Scene Designer and select Archive Door.
2. In Verb Actions, set Default Verb to Use for this context-sensitive test.
3. In Use Verb, choose **Run Visual Script**.
4. Choose **Check the Archive Door(check_door)** as Script.
5. Apply and save.

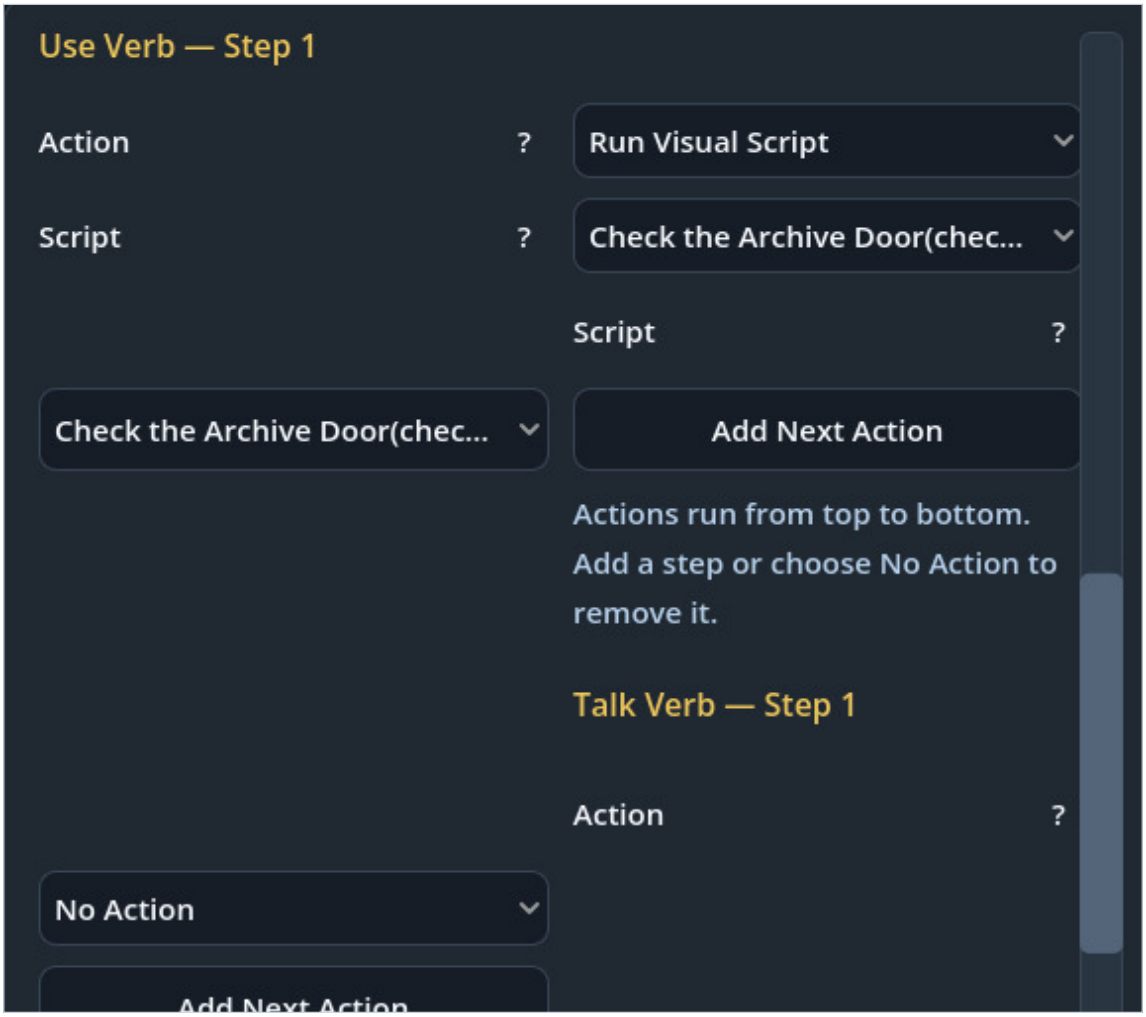

Figure 34. The door's ordinary action list calls the reusable visual script. The graph decides which response to run.

For this prototype test the door has auto-approach disabled. You can later enable an approach once you have built a real doorway and its walking route.

Test from a fresh run: use the door first, repair the clock, then use the door again. The first attempt should refuse; the second should take the success route. Do not set the designer's initial flag to true just to make the door appear to work.

19. Add an automatic entry hint

An event volume can react when the player enters an area. It does not need to be a visible prop or a solid obstacle.

1. Choose **Add to Scene > Trigger**.
2. Name it **Workshop Entry Hint**, with ID `entry_hint`.
3. Keep its Type as **Trigger**. It is an event area, rather than a visible wall.
4. Set its 3D Position to **0, 1, 0.3** and Scale to 1, 1, 1.
5. In General, use a Box Event Volume with Size **8, 2, 0.8**.
6. Keep its custom event-volume position off for this centred example.

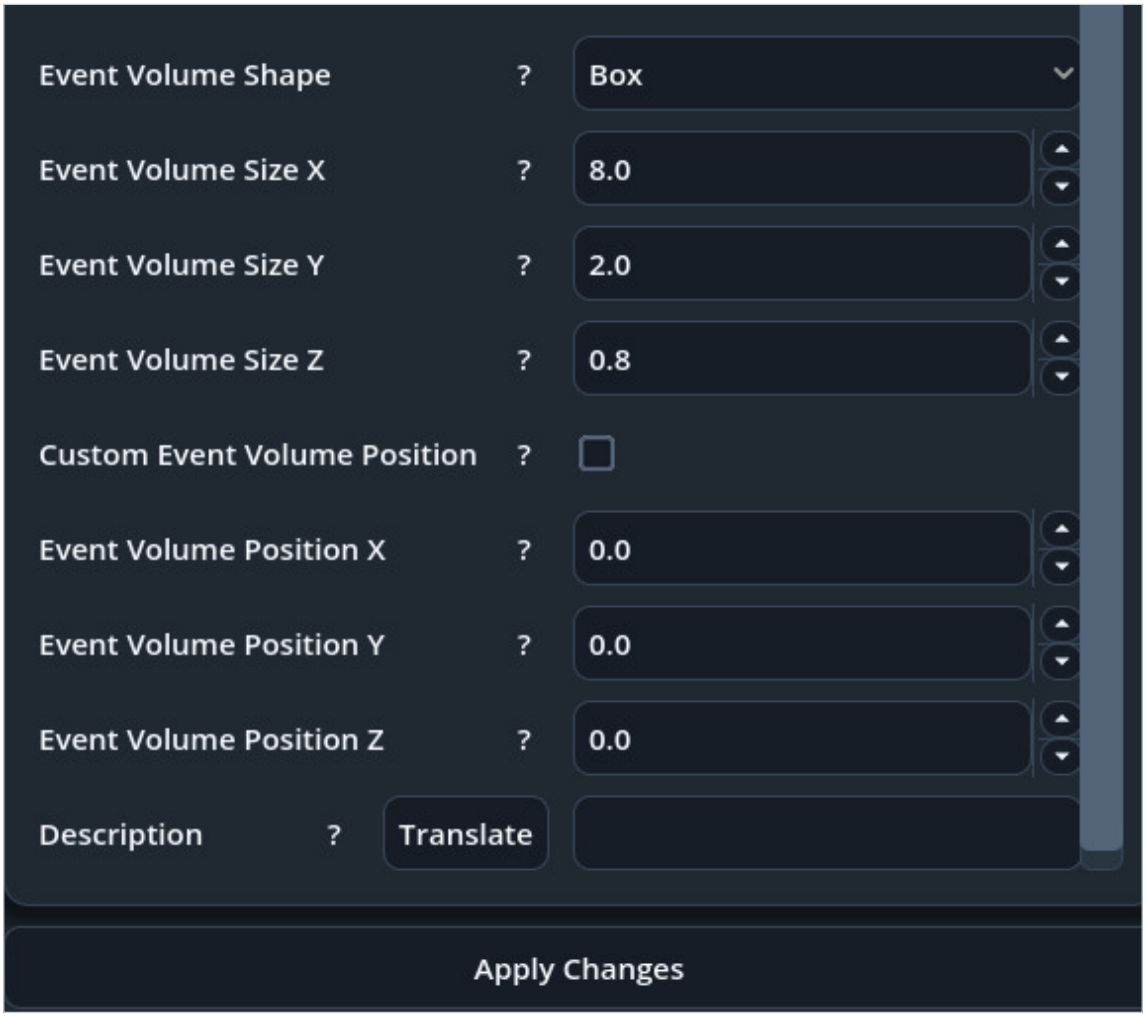

Figure 35. The trigger covers a strip across the room. The player's starting position is outside it.

Open **Trigger Events**. In **Player Collision / Entry**, add these actions:

Step	Action	Settings
1	Show Message	The clock is waiting for a small repair.; Duration 2; Block Player Input off.
2	Enable / Disable Trigger	Trigger: Workshop Entry Hint; Enabled off.

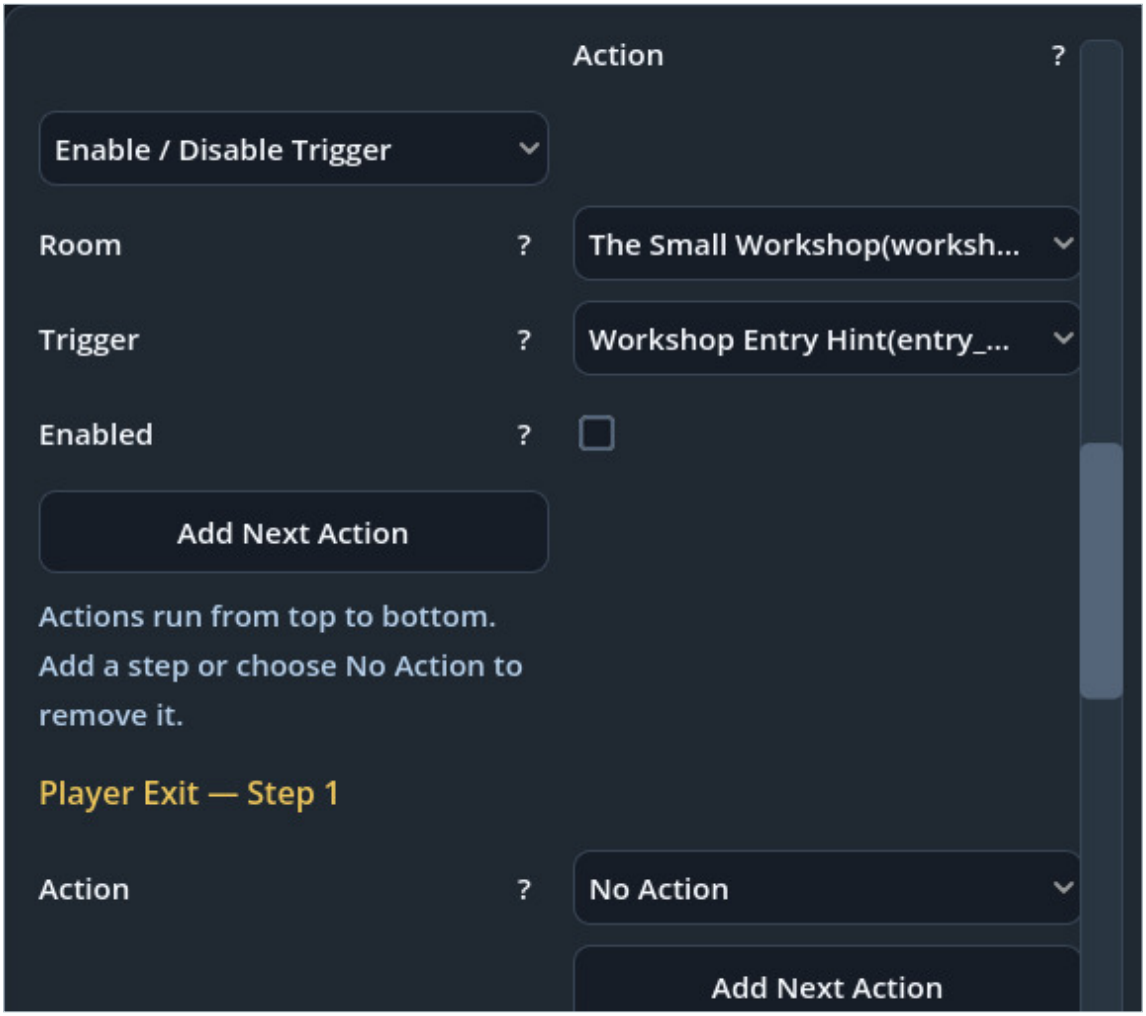

Figure 36. The second entry action disables this trigger, so walking back through it does not repeat the hint in the same run.

Entry and exit events concern crossing the volume. Left Click and Hover concern pointer interaction. Choose the event which matches the behaviour you want.

If the hint fires immediately when the room starts, the character may already overlap the volume. Move or resize it. If it never fires, check the trigger's enabled state, its vertical placement and whether the actual walking route crosses it.

20. Give the lantern a real light

A bright-looking lantern material and a room light are different things. Add an authored point light to illuminate the nearby table and floor.

1. Choose **Add to Scene > Point Light**.
2. Give it ID `lantern_light` and Name **Lantern Light**.
3. Place it at **-3.1, 1.7, -1.8** for this lantern.
4. In Light Settings, use a warm Light Color such as **#ffc879**.
5. Start Intensity at **1.2** and Range / Distance at about **3.5**.
6. Leave it on. Start with Cast Shadows off for this extra light; the directional room light already supplies the main shadows. Turn local shadows on later if the shot needs them.
7. Apply and test the game view.

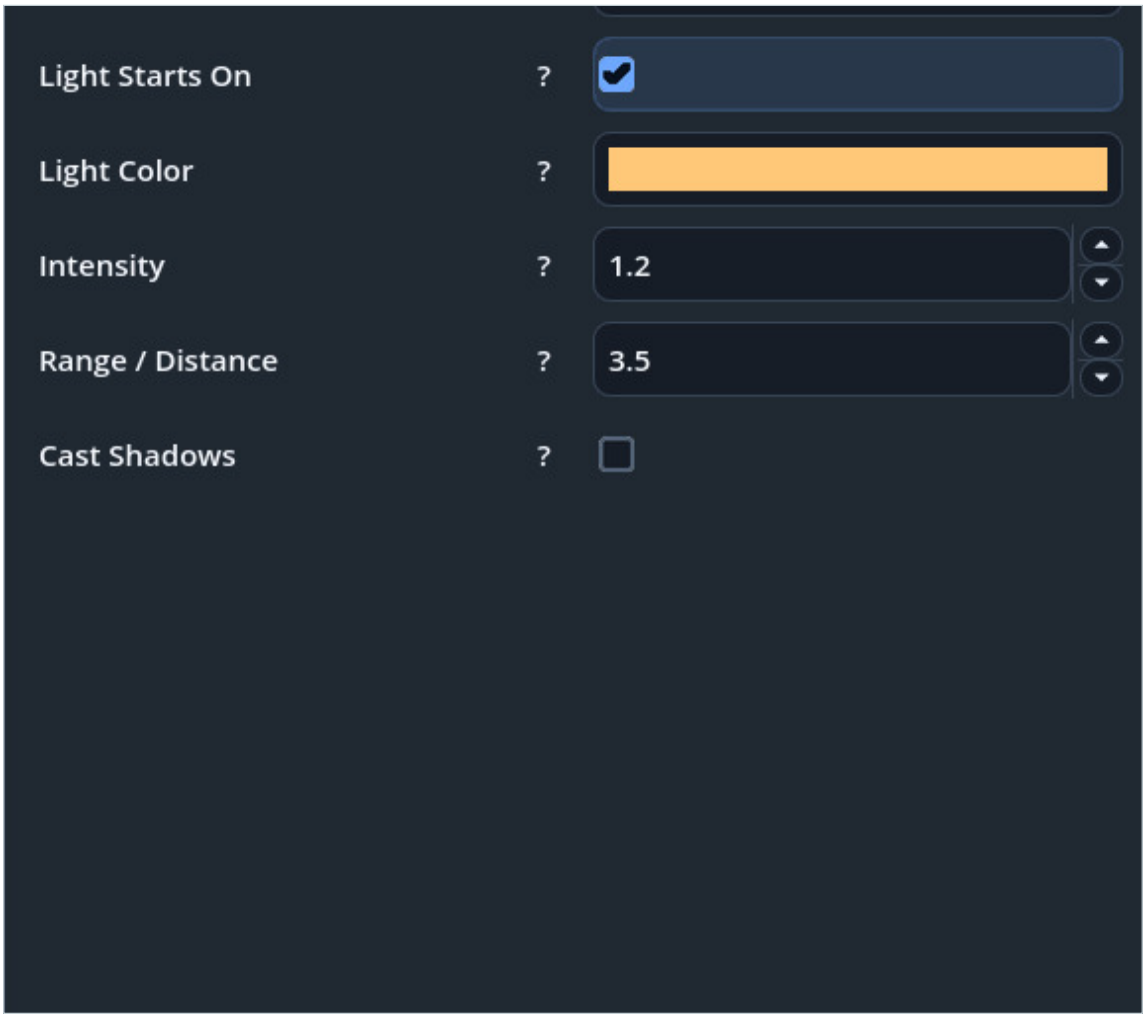

Figure 37. A point light has a position and range. It adds actual local lighting rather than only colouring the lantern's own material.

Keep the directional room light from the first guide as the broad fill. A point light is for a smaller area. Adding many strong lights is more likely to wash out the furniture than make it more attractive.

Emission can make a surface look bright. A visible halo, light falling onto other objects and a mirror image are separate effects. Check each result in the renderer you intend to use.

21. Add a local region effect

A region can mark part of the floor for interactions or local visual treatment. We will add a restrained warm tint around the workbench without changing the whole room.

1. Add a **Region**.
2. Give it ID `workbench_glow` and Name **Warm Workbench Area**.
3. In General, set Minimum X **-4.8**, Maximum X **-1.7**, Minimum Z **-2.8**, Maximum Z **-0.8**.
4. Keep Height Above Floor at 0 and Event Volume Height around **2.4**.
5. Apply the shape, then open Visual Effects.

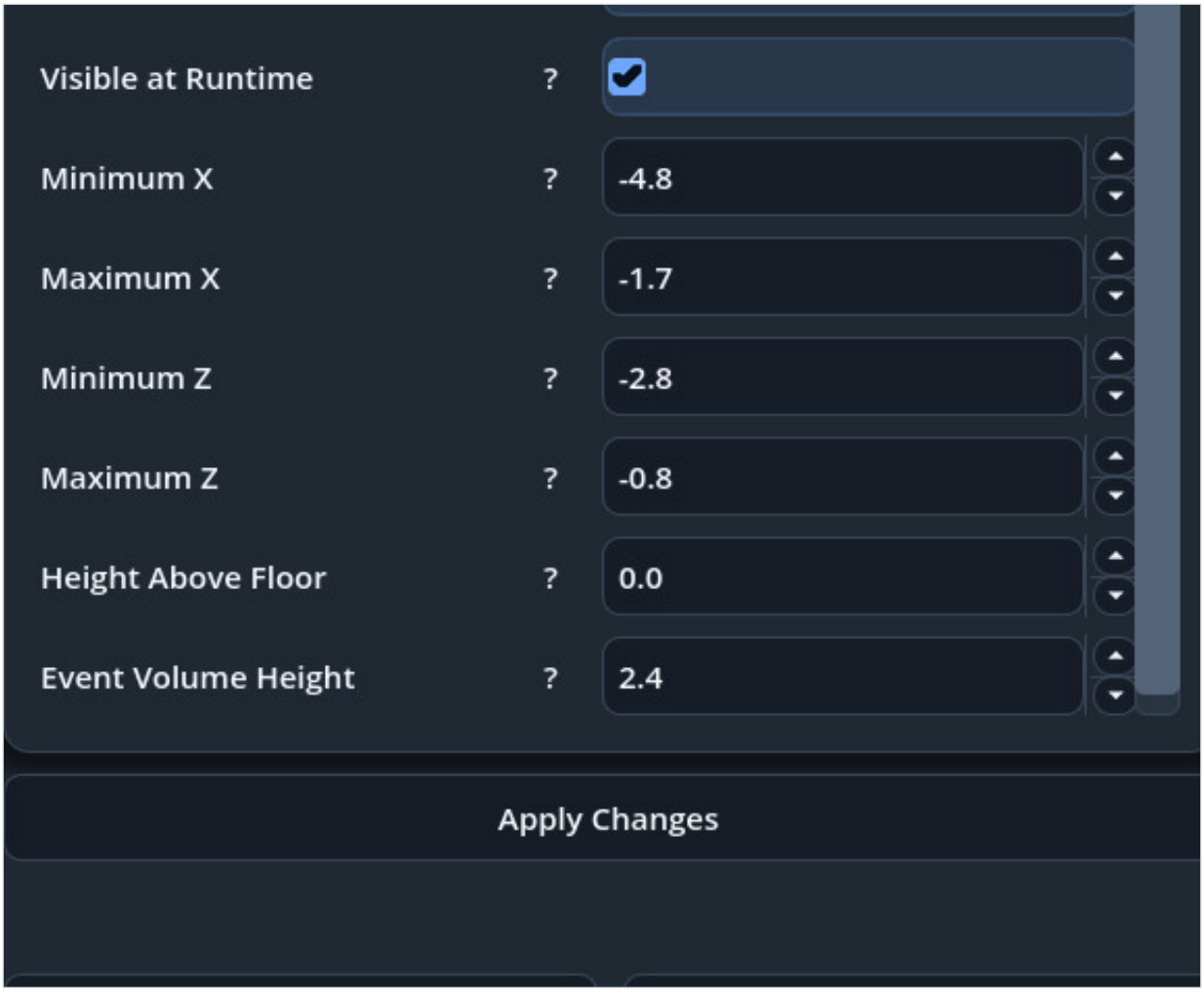

Figure 38. The region's footprint is separate from the walkable navigation area and from the table's collider.

Turn **Enable Region Effects** on. Set Tint Color to **#ffcf8c**, Tint Strength to about **0.12**, Blur Radius and Bloom Strength to 0, Preset to **Disabled** (none) and Surface Type to None.



Figure 39. A gentle tint affects the region footprint. This example deliberately adds no blur and creates no physical surface.

The screen treatment covers the region's footprint and objects drawn over it. It is not a replacement for the point light. Keep its strength modest so the art remains clear.

The same tab offers blur, bloom, built-in presets, custom screen shaders and surface treatments. A custom screen shader replaces the built-in screen effects. A material surface, mirror surface or custom surface shader is another layer of behaviour and needs its own settings. Test one treatment at a time.

A region also has Region Events and Verb Actions. Use them if the area needs an entry response or an interaction. Defining a visual tint does not automatically create those actions, and a region is not automatically a navigation blocker.

22. Add a small flame effect

This optional step adds a small visual flame near the lantern. It is not needed for the repair puzzle.

1. Add a **Particle Emitter**.
2. Give it ID `lantern_flame` and Name **Lantern Flame**.
3. Place it at **-3.1, 1.55, -1.8** for the pictured lantern.
4. Open Particle Settings and choose **fire** as Preset first.
5. After choosing the preset, reduce Amount to about **18**, Lifetime to **0.6**, and Particle Size to about **0.05**.
6. Keep Emit In Game on and One Shot off for a continuous flame.

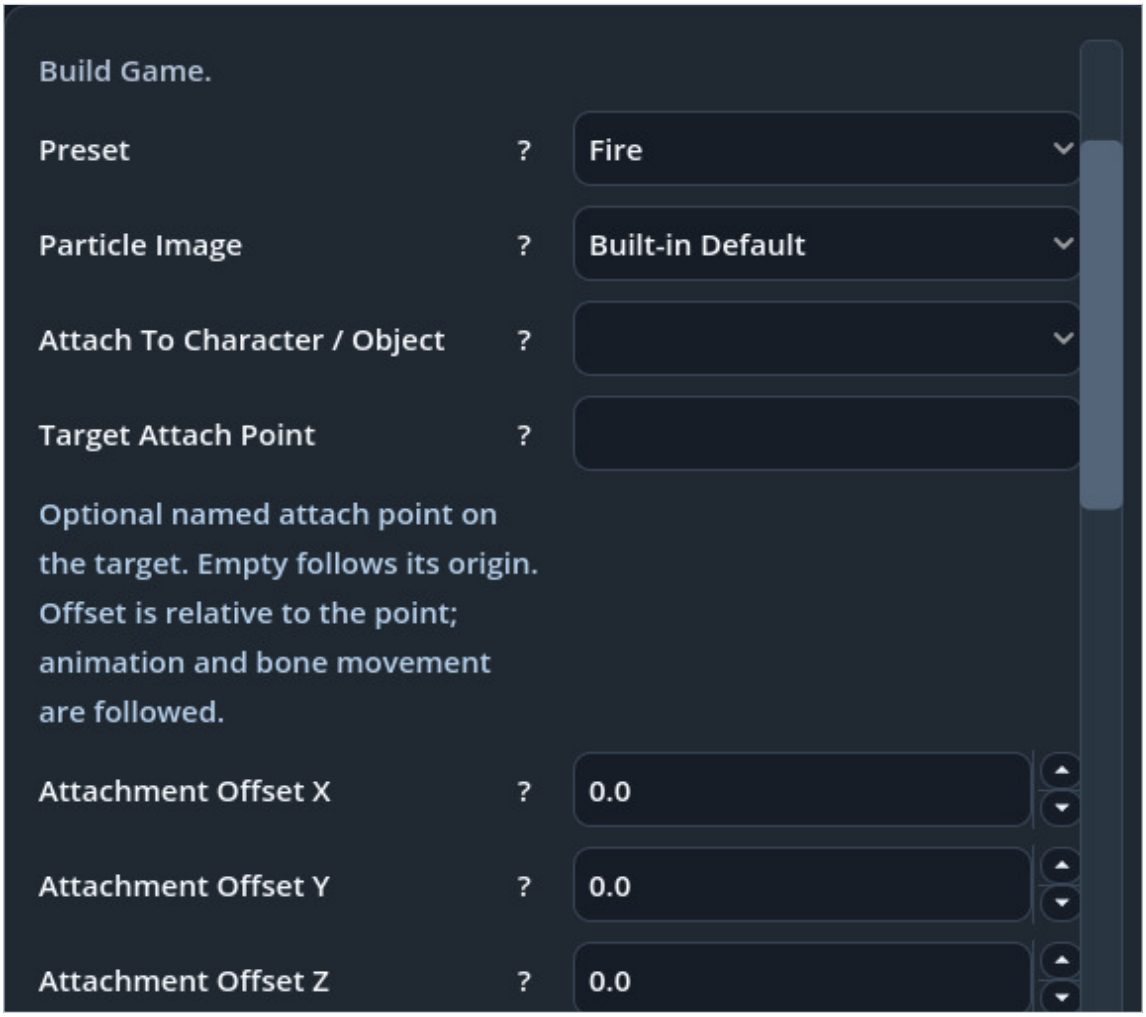

Figure 40. Choose a preset before fine-tuning. Changing the preset loads a new recipe and can replace adjustments you made earlier.



Figure 41. The smaller flame uses a modest particle count, lifetime and size. The preset is only the starting recipe.

In Motion & Emission, begin with **Velocity Min 0.1**, **Velocity Max 0.15**, Spread about **12**, and Turbulence Enabled off for a calmer small flame. Use Direction **0, 1, 0**, Gravity **0, 0.1, 0**, and Emission Area Extents **0.04, 0.02, 0.04**. Judge the result next to the model and reduce it if it obscures the lantern.

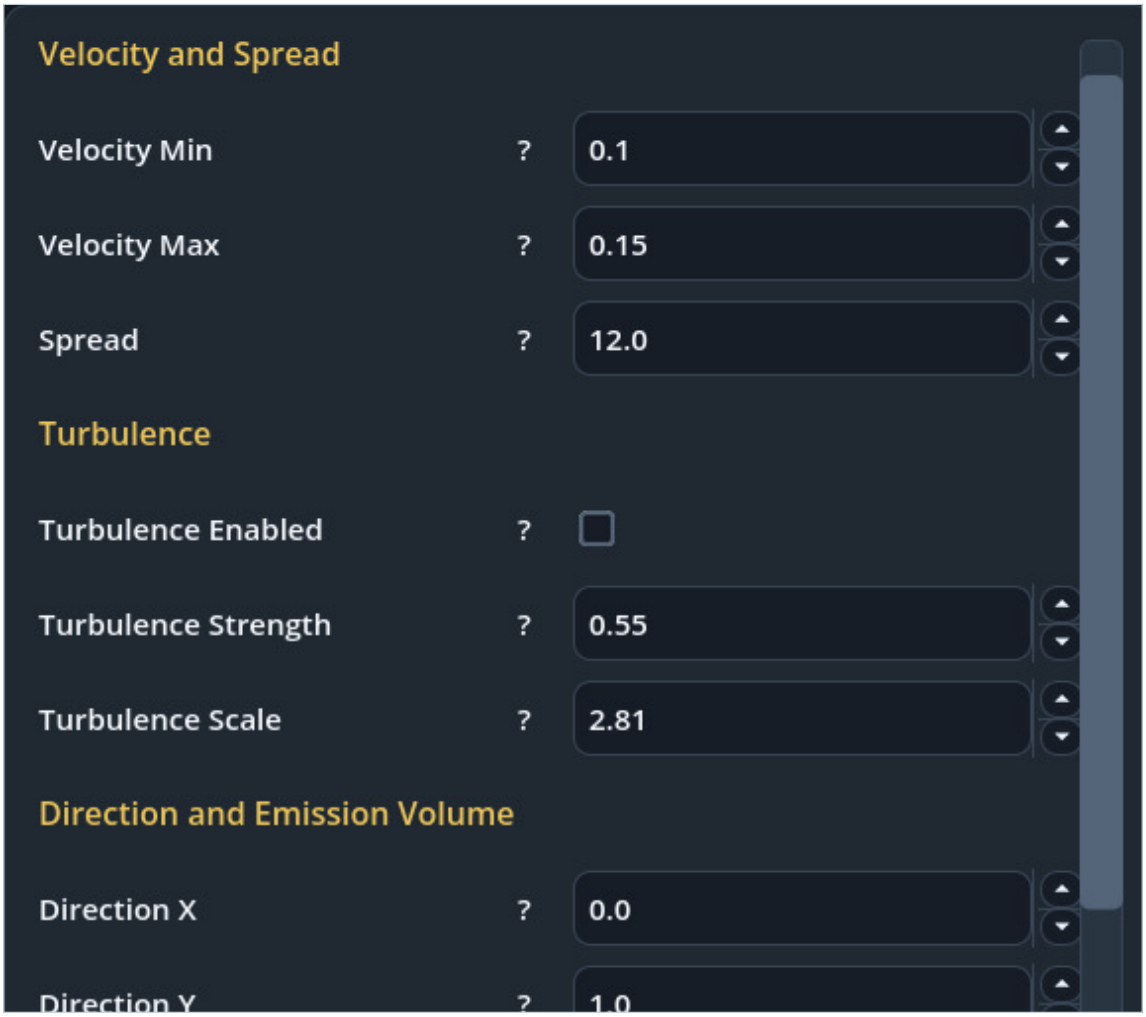

Figure 42. Velocity Min and Max control the initial speed range. A tiny flame needs much smaller values than a room-sized fire.

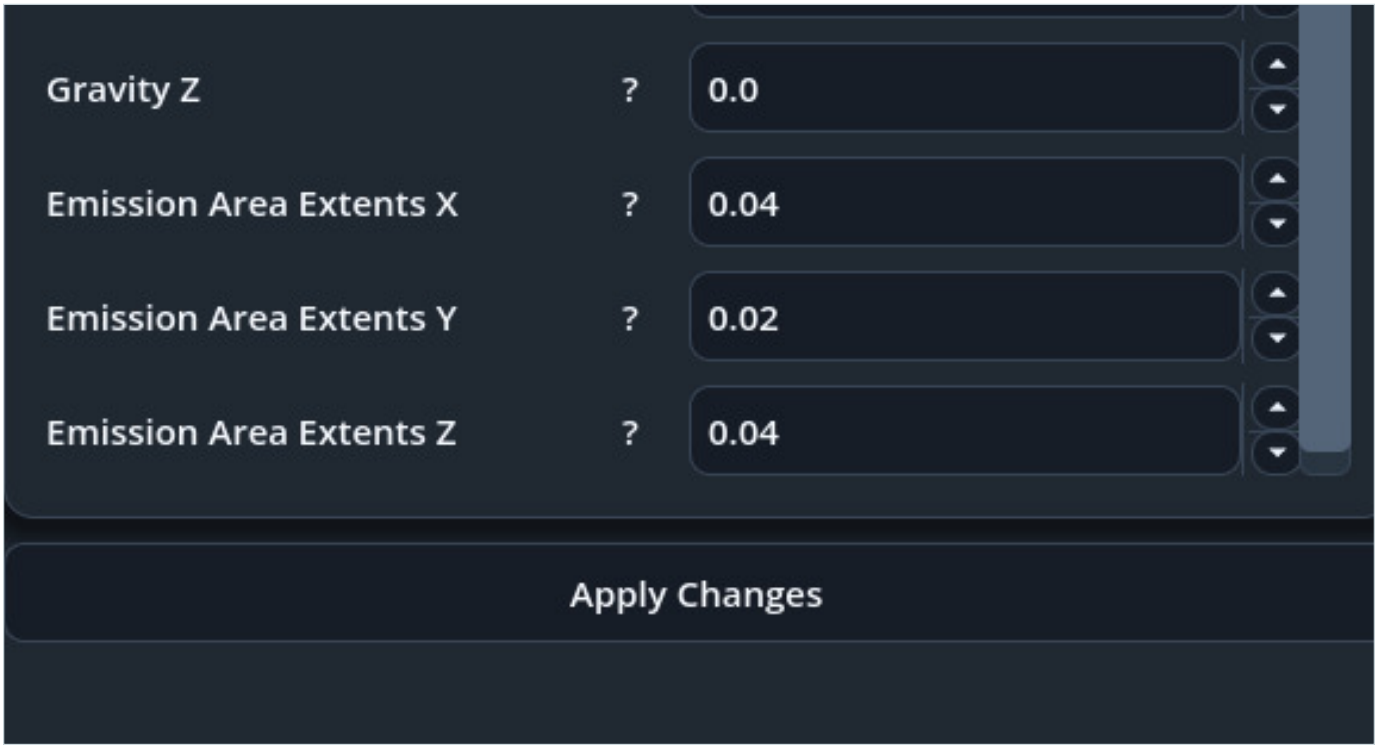

Figure 43. The emission area controls where particles start. It is not the event volume or the physical collider.

A flame effect does not automatically burn the player or play sound. Add events and audio deliberately if the game needs them. A large explosion, storm or snow preset also needs its own scale and placement; the tiny lantern values are not appropriate for those effects.

23. Test the scene as a whole

It is time to check how the systems behave together. Save before starting a fresh game preview.

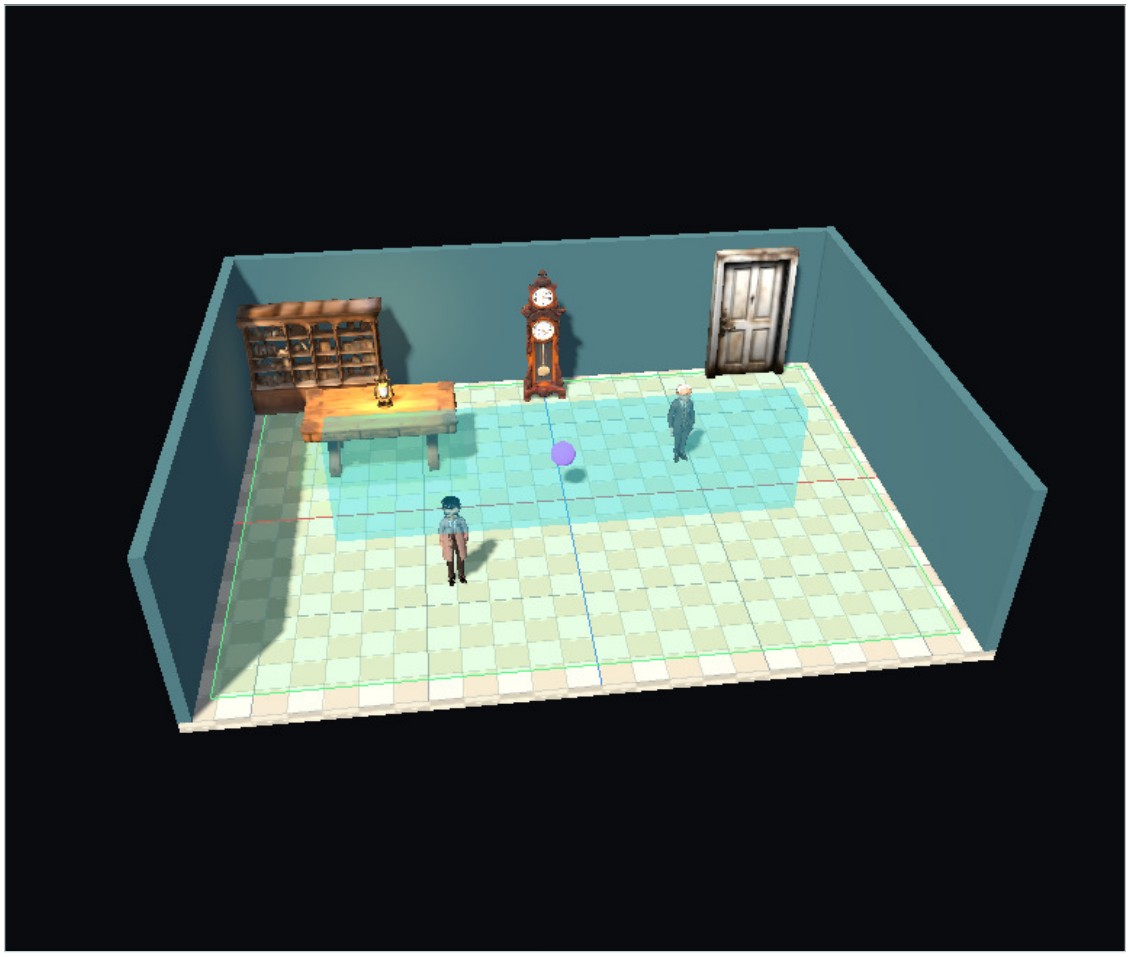

Figure 44. The completed scene in the editor, with imported art, characters, lights and authored gameplay objects.

Test	Expected result
Start the room	The normal wide camera opens and both character models are visible.
Walk around the workbench	The player uses its walk cycle and avoids the furniture.
Use the archive door before repairing	The refusal branch runs.
Cross the entry trigger	The hint appears once, then stops repeating.
Use the clock	The approach, camera, gesture, message, sound and variable actions run in order.
Finish the repair	Room View returns.
Use the door again	The true branch disables the door prop and shows success.

Test	Expected result
Talk to Mara	The timed speaking response runs with the correct character.
Stop and restart a new test	The initial false flag and trigger state are restored for a fresh run.

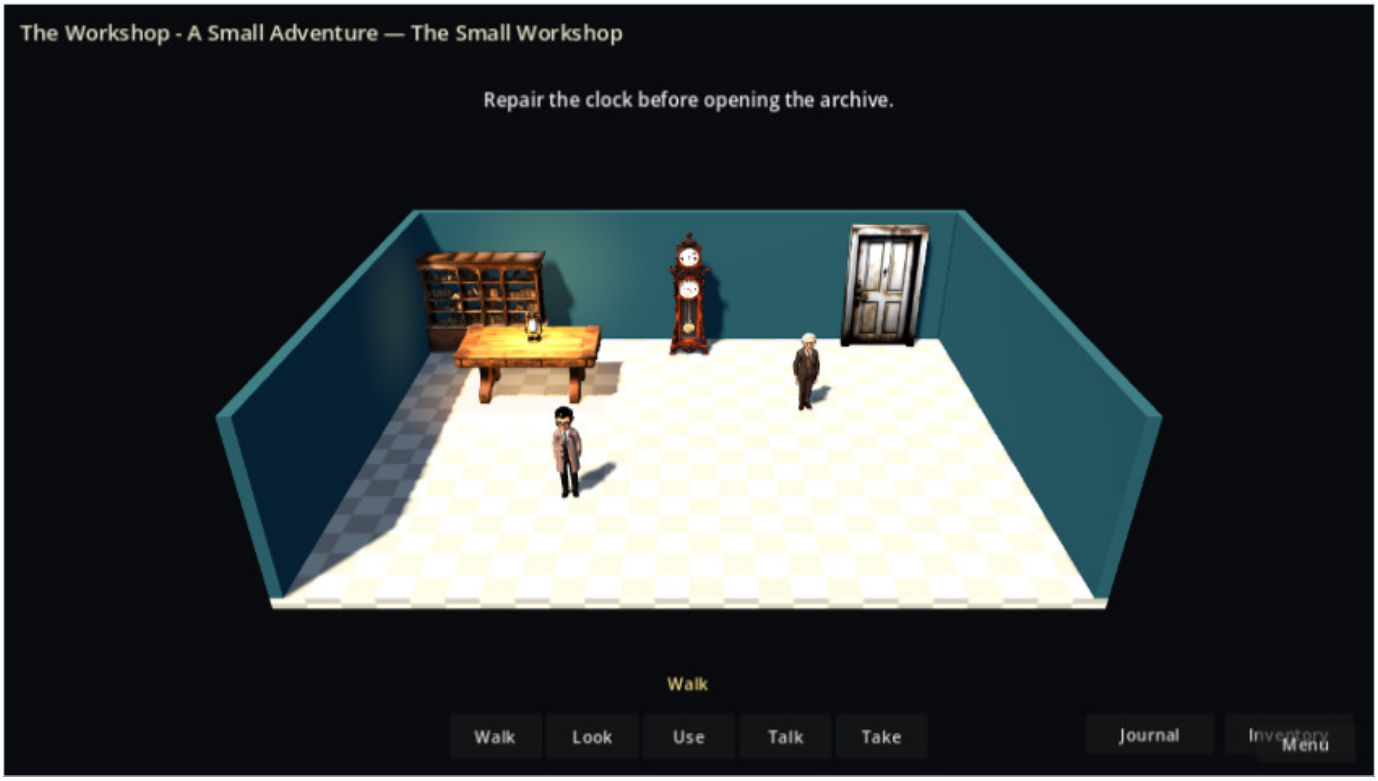

Figure 45. The door's refusal response before the clock has been repaired.

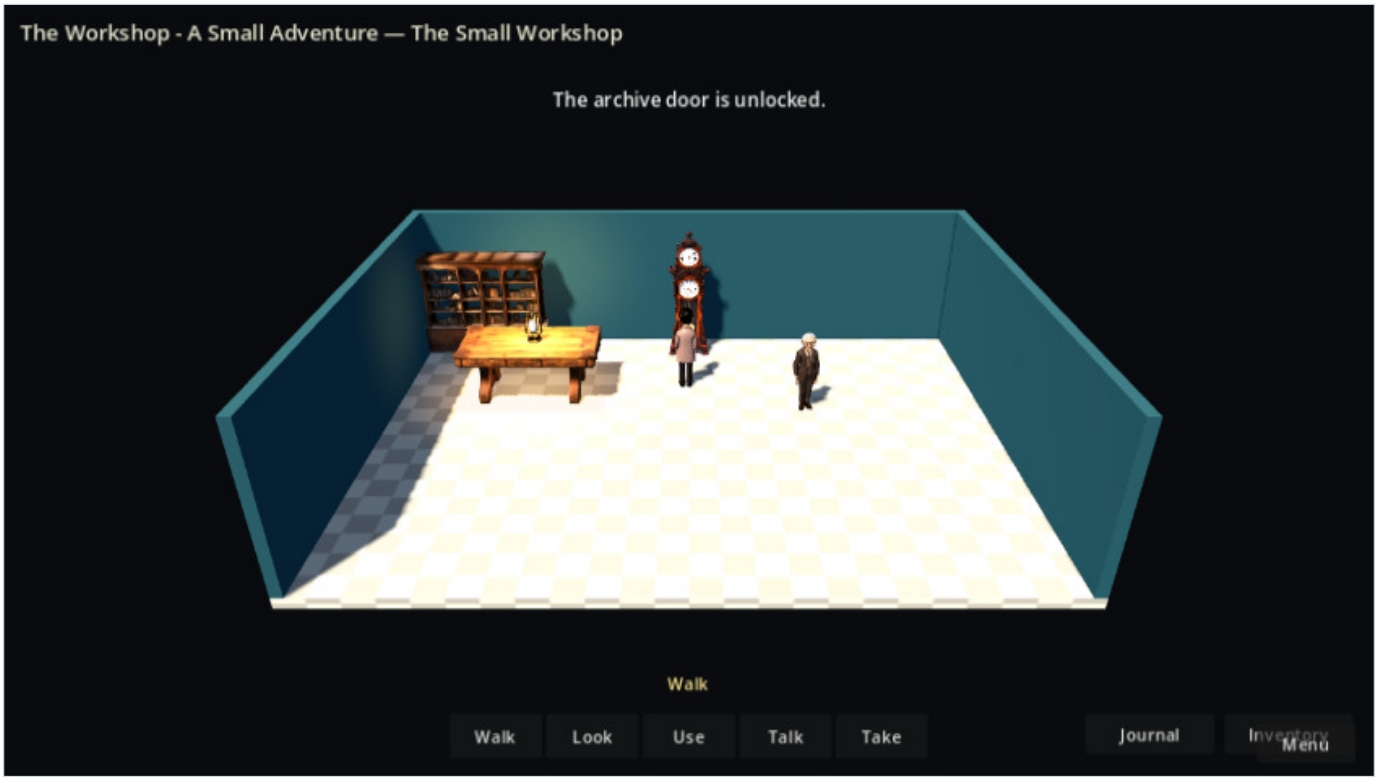

Figure 46. After the repair, the same door interaction takes the success route.

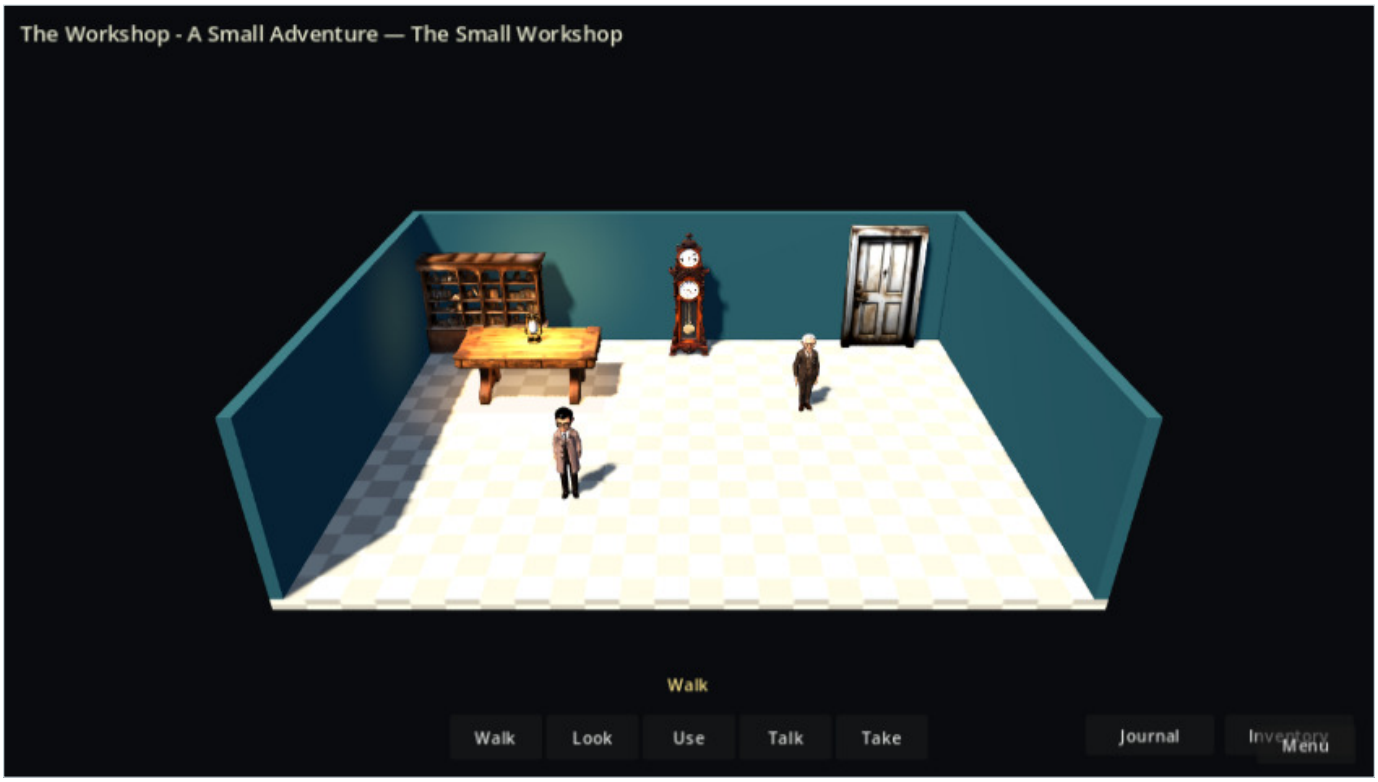

Figure 47. The finished scene in a fresh test run. The door is present again and the repair flag starts false, ready for another play-through.

Test Play / Restart too, not only Test Scene. The first checks project startup; the second checks the room currently being edited. Save/load tests are a separate useful pass once the ordinary new-game route works.

24. Fix the common problems

Problem	What to check
Prop still looks like a box	Type Asset, the Model / Image / Sprite Sheet reference, applied changes and the source asset preview.
Model is half under the floor	Its source origin, Position Y and uniform scale.
Character remains a capsule	The reusable Character model and the room instance's Character selector. Refresh an older placement from the updated definition if necessary.
Model appears but never walks properly	Model Clip source, the walk state ID, its chosen clip, Loop and Playback Speed.
Character drifts out of its body	Root motion, model orientation and the imported clip itself.
NPC uses the player's appearance or clip	Animation / Sprite Asset in copied states.
Imported colours are muddy	Model Tint and material overrides; start with white tint.
Player stops too far from furniture	Physical collider and character radius before Stop Distance.
Clock click runs the wrong response	Default Verb, the chosen interaction mode, the correct verb group and pointer-event overrides.
Camera stays close after the repair	The explicit later camera action restoring wide.
Flag changes but the door still refuses	Boolean true versus text, matching variable ID, graph connections and whether the test was restarted.
Door art disappears but the route is still closed	The wall, navigation and collision behind it. A hidden prop does not create a doorway in architecture.
Hint repeats on every pass	Its second action must disable the correct trigger.
Region tint affects more than expected	Its footprint and strength; screen treatment can include objects drawn over it.
Particle edits seem to reset	Choose the preset first, then tune its recipe.
Everything becomes slow after adding art	Test one model at a time, inspect texture sizes and mesh complexity, and reduce costly effects and overlapping lights.

Keep a useful order when building the next scene

Get one model visible. Get one character walking. Get one object response working. Then extend the action sequence, add a condition, and improve the lighting and atmosphere.

That order leaves you with a scene you can test at every stage. The room's art, navigation, characters and behaviour become easier to manage when each part has a clear job.

R1. The important settings at a glance

Setting	Example value
Player source asset	player_model, Kind Model.
NPC source asset	keeper_model, Kind Model.
Character height / radius	About 1.8 / 0.35 for these models.
Character state IDs	idle, walk, run, talk, inspect, nod.
Default character state	idle.
Character material tint	White to preserve source artwork colours.
Clock object / source asset	clock / clock_model.
Clock approach	Always Walk to This Object; Stop Distance about 1; face on.
Repair flag	clock_fixed; Boolean; initial false.
Ordinary camera	wide.
Close camera	clock_close; Position 0, 3, 1.4; Target 0, 1.5, -3.7; FOV 40.
Camera blend time	0.7 seconds; Wait for Blend on.
Door's script	check_door, called by Run Visual Script in Use Verb.
One-time hint trigger	entry_hint; Position 0, 1, 0.3; Box Size 8, 2, 0.8.
Warm point light	lantern_light; Position -3.1, 1.7, -1.8; Intensity 1.2; Range 3.5.
Warm region	X -4.8 to -1.7; Z -2.8 to -0.8; Tint Strength 0.12.
Optional flame	fire preset; Amount 18; Lifetime 0.6; Particle Size about 0.05.

The pictures and settings document the named engine version. The scene was built and checked in a running copy of the editor. No project, model pack or practice files are included with the PDF.