PCA 3D ADVENTURE STUDIO

The Cutscene Designer

From the first shot to the return to gameplay.



Figure 1. The Silent Bell running in PCA. This guide builds the complete introduction and its gameplay handoff.

A detailed, illustrated guide to timeline editing, camera work, character staging, speech, effects, skipping and cleanup.

For PCA 3D Adventure Studio 0.86.3.4

How to use this guide

Build one small, working sequence first. Then add its staging, give it a finished interface and make its normal and skipped endings reliable.

Lessons 1-14 explain the workspace and the timeline’s timing rules. Lessons 15-28 build the workshop scene, from its shot list through movement and recorded speech. Lessons 29-42 add captions, audio, interfaces, effects, scripts and movies.

Lessons 43-56 connect the cutscene to the game and cover skipping, cleanup, testing, the current Clockmaker example and the release pass. The appendices give the complete timeline record and the short scripts used around it.

The most important distinction

A timeline event starts at a time. It does not automatically wait for the previous event to finish. The duration of a block is also not a universal “stop this action” instruction. Those two rules explain most of the surprises people meet while learning this designer.

Screenshots and examples

All editor and Play illustrations are actual native captures. The small workshop is deliberately simple so the controls and actors are easy to follow. The characters and sample performances come from the bundled Clockmaker content.

The instructions apply across desktop platforms. Panel positions and visible toolbar buttons depend on screen size, theme and UI scale. Horizontal scrolling exposes controls that do not fit.

No practice project is included or required. Follow the instructions in a project of your own, using your own IDs where needed. Save a copy before editing an existing cinematic.

Code and JSON

Copy JSON with its braces, commas and ordinary double quotes intact. GDScript indentation matters. The scripts use PCAScriptBase and PCA’s action API; they do not require changing the engine source.

Contents

1. What a cutscene controls . . . 5
2. The scene we will build . . . 6
3. Create and open a cutscene . . . 7
4. Find your way around the workspace . . . 8
5. Set the cutscene options . . . 9
6. Read an event before editing it . . . 10
7. Understand the small amount of JSON you need . . . 11
8. Add the first camera event . . . 12
9. Apply the event, then save the project . . . 13
10. Use the playhead as an editing position . . . 14
11. Move, trim and snap event blocks . . . 15
12. Zoom, sort and set the ending . . . 16
13. Plan parallel work and waiting separately . . . 17
14. Test a small sequence before expanding it . . . 18
15. Make a shot list before adding every block . . . 19
16. Prepare the room and actor marks . . . 20
17. Author named cameras in the room . . . 21
18. Cut to a named shot or set a raw pose . . . 22
19. Move the camera over time . . . 23
20. Blend between authored cameras . . . 24
21. Follow a moving actor or frame a group . . . 25
22. Change targets, priorities and orbit settings . . . 26
23. Move a character to a mark . . . 27
24. Turn actors toward each other . . . 28
25. Use animation states and imported clips . . . 29
26. Schedule a timed spoken line . . . 30
27. Add the recording and mouth timing . . . 31
28. Leave enough room for speech and typewriter text . . . 32
29. Show a caption, name or changing value . . . 33
30. Put sounds on named channels . . . 34
31. Keep voice playback separate from sound effects . . . 35
32. Build a readable subtitle interface . . . 36
33. Make the opening title a GUI . . . 37
34. Understand automatic restoration . . . 38
35. Use restrained camera and screen shake . . . 39
36. Change lighting, tint, glow and motion . . . 40
37. Add screen effects without obscuring the scene . . . 41

38. Call a script from an event . . . 42
39. Call exposed functions with parameters . . . 43
40. Finish the cutscene before opening choices . . . 44
41. Use a movie or recorded playback deliberately . . . 45
42. Props, attach points and other advanced events . . . 46
43. Choose the real launch point . . . 47
44. Decide what “play once” means . . . 48
45. Understand exactly what skip does . . . 49
46. Wrap playback in one cleanup route . . . 50
47. Add a mouse or touch skip button . . . 51
48. Establish the same useful final state . . . 52
49. Use the right test for each job . . . 53
50. Run a completion and skip checklist . . . 54
51. Read the bundled Clockmaker introduction . . . 55
52. Make the staging easy to follow . . . 56
53. Fix timing and flow problems . . . 57
54. Fix missing cameras, actors and media . . . 58
55. Check languages, controls and performance . . . 59
56. Finish the release pass . . . 60
Appendix A. Complete workshop timeline . . . 61
Appendix B. The guarded launch script . . . 68
Appendix C. Timing and ownership reference . . . 71
Appendix D. The complete event schedule . . . 72
Version and verification notes . . . 73

1. What a cutscene controls

A cutscene is a set of events placed on a clock. At a chosen time, PCA can move a camera, send a character across the room, play a line, show a title or change something in the game.

The Cutscene Designer is useful when several things must happen together. While Ivo walks toward Elias, the camera can move closer. A reply can begin after the walk, followed by a reverse angle and a return to normal gameplay.



Figure 2. The workshop exchange running in Play. The camera, speech and character events come from one timeline.

Use this tool	For this job
Cutscene Designer	Time-based staging: shots, movement, speech and effects.
Dialogue Designer	Questions, branching responses and conversations controlled by the player.
Action list / script	Work that must wait for an earlier step or check a condition.
GUI Designer	The appearance of titles, subtitles, skip controls and other overlays.

> The timeline is a schedule, rather than a recording of everything in the room. You author actions; PCA performs them when the game runs.

2. The scene we will build

The example is The Silent Bell: an 18-second introduction in a small workshop. Elias notices that the usual bell has not rung. Ivo approaches, answers him, and the camera returns to a playable room.

Record	Example ID	Purpose
Room	lab	A simple 3D workshop with a walkable floor.
Active player	ada / player	Professor Ivo; player is the runtime alias.
NPC	elias	Elias, the clockmaker.
Cutscene	workshop_intro	The 18-second timeline.
Boolean variable	intro_seen	Starts false; prevents a shipped introduction repeating.
Title GUI	title_card	A short opening title.
Subtitle GUI	cinematic_speech	Speaker name and spoken text.
Sound	chime	A short accent on a named sound channel.

The figures use the bundled Clockmaker characters and recordings in a simple room. You can use other models and your own voice files. Replace every example ID with the ID in your project when the records differ.

> This PDF uses PCA 3D Adventure Studio 0.86.3.4. It is platform-neutral and contains no separate practice files. The editor and Play illustrations are actual captures.

3. Create and open a cutscene

1. Open Adventure Database → Cutscenes, or use the Cutscene Designer button in the designer toolbar.
2. Choose Add New to create a record, then give it a useful display name and a stable ID.
3. Select that record to open its timeline.
4. Set Name to The Silent Bell - Workshop Introduction. Use workshop_intro as the example record ID.
5. Save the project before building a longer sequence.

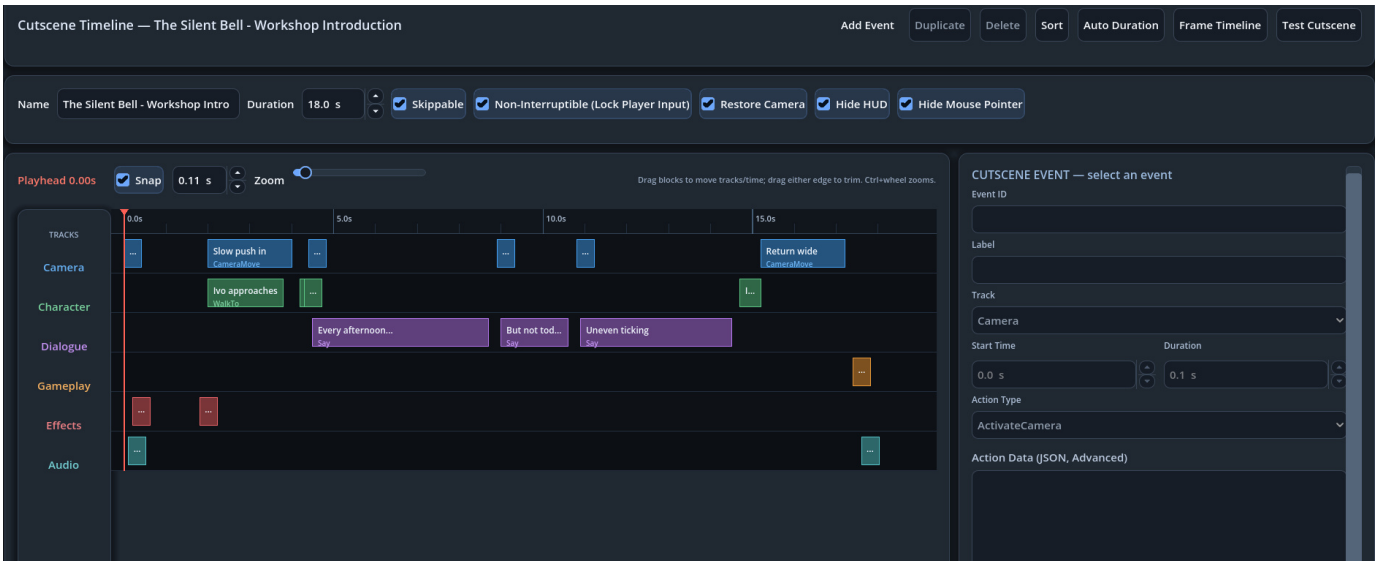

Figure 3. The timeline belongs to the selected cutscene record. The outer selector switches between complete cutscenes.

A new cutscene starts with an empty events list and a short declared duration. It will not show a movie or animate the room until you add events. Add New creates another whole cutscene; Add Event adds a step to the selected one.

> The cutscene ID is the reference used by Start Cutscene actions and scripts. Renaming the display title does not require changing that ID.

4. Find your way around the workspace

Area / control	What it does
Record selector	Chooses the complete cutscene. Add New and Clone Selected work at record level.
Add Event	Adds an action template at the current playhead time.
Duplicate / Delete	Copies or removes the selected event.
Sort	Orders stored events by time, then track for equal times.
Auto Duration	Calculates a duration from the last event, with a small ending margin.
Frame Timeline	Fits as much of the sequence as the zoom range allows.
Test Cutscene	Restarts preview and runs the selected cutscene.
Settings bar	Name, total duration, skip, input, camera, HUD and pointer options.
Playhead / Snap / Zoom	Controls where new events go and how you edit timing.
Timeline lanes	Shows events arranged by time and authoring track.
Event inspector	Edits the selected event and its action data.

There are two different inspectors to keep straight. The workspace’s event inspector edits the selected block. The main docked Inspector can show the whole cutscene record, including Raw JSON. Make more room by hiding the main Inspector when you only need event editing.

> On a narrow screen, toolbar controls can extend sideways. Use their horizontal scrolling or arrows before assuming an option is missing.

5. Set the cutscene options

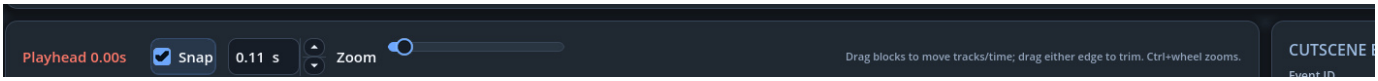

Figure 4. The settings bar controls how the cutscene temporarily takes over from gameplay.

Option	Behavior
Duration	The declared length in seconds. Later event ends can extend the actual runtime span.
Skippable	Allows the normal skip request, including Escape.
Non-Interruptible (Lock Player Input)	Blocks ordinary gameplay input beneath the cutscene. It does not override Skippable.
Restore Camera	Restores the pre-cutscene camera state on normal finish or skip.
Hide HUD	Temporarily hides the gameplay HUD and restores its previous visibility afterward.
Hide Mouse Pointer	Temporarily hides the pointer and restores its previous mouse mode afterward.

For the example, set Duration to 18 seconds and turn on all five checkboxes. The player can skip deliberately, but cannot issue a new Walk command while the actors are being staged.

> Restoration is specific. Restore Camera does not undo character movement, inventory changes, lighting changes or every GUI you displayed. Those need their own plan.

6. Read an event before editing it



Figure 5. The common event fields identify one block and place it on the timeline.

Field	Meaning
Event ID	A unique reference inside this cutscene.
Label	The readable description on the block.
Track	Its authoring lane: Camera, Character, Dialogue, Gameplay, Effects or Audio.
Start Time	When the event begins, in seconds from the start.
Duration	The block’s span. Its runtime meaning depends on the action.
Action Type	The action template or structured action fields.
Action Data (JSON, Advanced)	The actual action object that playback uses.

A label is a reminder to you. Changing “Ivo approaches” to “Ivo leaves” does not change the character’s destination. Edit the action data as well.

7. Understand the small amount of JSON you need

Many traditional cutscene actions use the Action Data box for their settings. Advanced camera, rendering and exposed-function actions also have dedicated fields. You do not need to write a program to edit a simple JSON action.

```
{
  "type": "SetCamera",
  "position": [0, 4.5, 9],
  "target": [0, 1, -0.3],
  "fov": 42
}
```

Piece	Read it as
"type": "SetCamera"	The action to perform. Keep the spelling.
"position": [0, 4.5, 9]	A three-number X, Y, Z position.
"target": [0, 1, -0.3]	The point the camera looks toward.
"fov": 42	The perspective field of view, in degrees.
true / false	Boolean values; no quotation marks.
Comma	Separates entries; do not add one after the final entry.

Keep the outer braces. Put double quotation marks around field names and text values. Numbers stay unquoted. Use a decimal point, rather than a decimal comma, inside the data.

> A valid JSON object is not automatically a valid shot. A misspelled character ID can still look perfectly correct as JSON. Apply the event, validate the project and test the result.

8. Add the first camera event

1. Click the ruler at zero seconds.
2. Choose Add Event → SetCamera.
3. Set Event ID to establish and Label to Workshop wide.
4. Choose the Camera track. Set Start Time to 0 and Duration to 0.1.
5. Replace Action Data with the SetCamera object on the previous page.
6. Choose Validate and Apply Event, then Save Project.

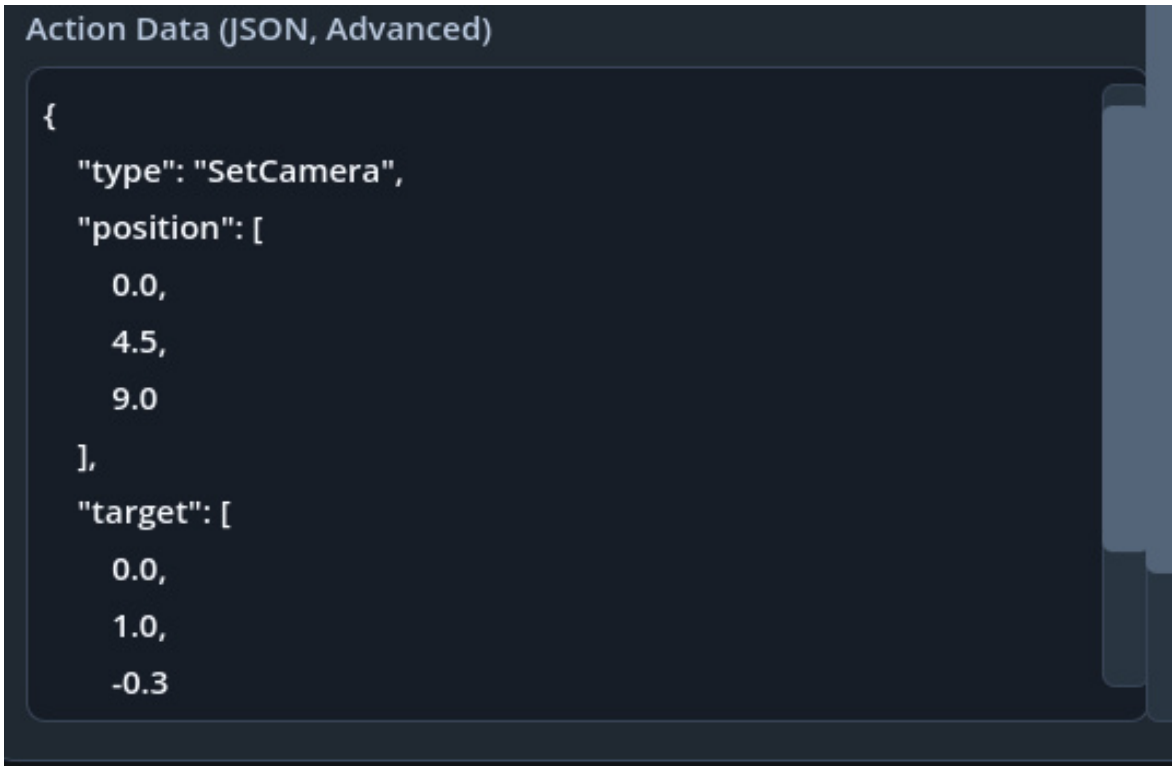

Figure 6. This event sets a complete camera pose at the beginning of the sequence.

The action sets position, look target and FOV immediately. The 0.1-second bar is a useful short event marker; it does not mean the camera automatically returns after one tenth of a second.

> Start with one camera event in a short test. Confirm the view before adding walking, subtitles and effects.

9. Apply the event, then save the project

Event fields are a draft until you choose Validate and Apply Event. That button transfers the event edits into the cutscene record. Save Project then writes the project to disk.

Problem	What to check
Event ID cannot be empty	Enter a non-empty ID.
Another event already uses that ID	Give the copy its own ID.
Action Data must be a valid JSON object	Check braces, commas, quotes and the outer object.
A change disappears after selecting another event	Apply the event before moving on; resolve any pending-change prompt deliberately.
The wrong action runs	Check the type value inside JSON, not only the dropdown caption.

Changing Action Type loads a new default action object. It can replace the parameters you entered for the previous type. Copy useful values elsewhere before changing types on an event you want to keep.

An Apply success message confirms the editable object was accepted. It does not prove that every asset, camera, target or animation exists. Project validation and Play answer those questions.

> If you edit the JSON type directly, keep the dropdown and data in agreement. Runtime dispatch follows the action’s type field.

10. Use the playhead as an editing position

Click the time ruler to move the red playhead. Add Event uses that time for the new event. Clicking empty timeline space can also move the playhead and clear the selection.

The playhead is an authoring marker in this version. Moving it does not reconstruct the room at that instant, seek every sound or preview the middle of a character animation.

You want to…	Use
Add a reply at 9 seconds	Move the playhead to 9, then Add Event.
Change an existing event precisely	Select it and edit Start Time in its inspector.
Watch the sequence	Test Cutscene.
See only a later moment while developing	Make a temporary short test cutscene or shorten a copied sequence, including the setup it needs.

Test Cutscene runs the selected record from its start. It does not start at the current red line. A later close-up may rely on earlier movement, so testing a copied late section needs a known starting pose.

> Do not confuse this timeline with a video editor’s frame-by-frame scrubbing. Movie playback has its own seek controls, covered later.

11. Move, trim and snap event blocks

Gesture	Changes
Drag the middle horizontally	Start time; preserves duration.
Drag the middle vertically	Track; preserves the action type.
Drag the left edge	Start and duration, keeping the old end in place.
Drag the right edge	Duration, keeping the start in place.
Snap enabled	Timing edits align to the selected interval.

Use a 0.1-second snap interval while arranging the example. Turn Snap off or use a smaller interval for a finer adjustment. The numeric Start Time and Duration fields are useful when the block is too short to grab comfortably.

Trimming synchronizes CameraMove and CameraShake durations, Say timeout and ShowMessage duration. It does not crop the source audio file, remove frames from a model clip or stop a WalkTo at the bar’s right edge.

Duplicate creates a new event ID and places the copy after the original start by at least its original duration or 0.1 seconds. Inspect the copy’s target and time before applying it to a different actor.

> Tracks organize the screen. A sound placed on Gameplay still plays as a sound. Move it to Audio for clarity; doing so does not add audio-specific trimming or stop behavior.

12. Zoom, sort and set the ending

Control	Use
Zoom slider / Ctrl + wheel	Change seconds-to-screen scale.
Horizontal scrollbar	Reach later times in a long sequence.
Frame Timeline	Fit the available timeline width as far as the zoom limit allows.
Sort	Store events in time order and keep the selected event identified.
Auto Duration	Set the declared duration from the greatest event end, plus an ending margin.
Duration in settings	Keep a deliberate quiet hold after the last event.

The runtime also considers event ends. Reducing Duration below a later block’s end is not a way to cut that block off. Move, shorten or delete the event itself, then use Auto Duration and review the result.

The editor normally extends the cutscene when you add, move, trim or apply an event beyond its old end. Deleting a late event does not automatically shorten the whole sequence; use Auto Duration when you want that.

> Avoid depending on the order of two meaningful state changes at exactly the same timestamp. Give dependent operations distinct times, or put them in one sequential script.

13. Plan parallel work and waiting separately

The timeline dispatches events when their start times arrive. It does not wait for each action’s asynchronous work to finish before dispatching the next scheduled event.

At this time	Event	What happens alongside it
2.0 s	CameraMove, 2 seconds	The camera moves while other events run.
2.0 s	WalkTo, displayed as 1.8 seconds	Ivo starts moving; the next event does not wait for arrival.
4.5 s	Say, 4.2 seconds	Speech begins; a later scheduled event can still start.
Inside a called script	A wait / awaited action	That script waits; the timeline’s other events continue.

This is why a moving shot can accompany a walk. It is also why placing a long dialogue or movie event before a later action does not automatically postpone that action.

Use a sequential action list or script outside the timeline when the rule is “wait for this cutscene, then open a branching conversation.” Keep time-based staging inside the cutscene.

> One camera can only show one resulting view. Parallel work is useful for different jobs; two overlapping camera moves can compete over the same camera.

14. Test a small sequence before expanding it

1. Save and apply the first camera event.
2. Choose Test Cutscene.
3. Check that the intended room and character are present.
4. Watch the camera view, then check the handoff after the short sequence ends.
5. Return to the designer and add one more event at a time.

A cutscene does not own a separate private room. Its actor and camera IDs must exist in the room where playback begins. Test Cutscene restarts the game preview; check the resulting room rather than assuming it uses whichever scene you most recently looked at in Scene Designer.

For the example, lab is the player’s starting room. For a cutscene in another room, enter that room through a test route before starting it, or use an explicit room-loading sequence that you have tested.

> Test Scene starts an isolated room workflow. Test Cutscene checks a selected cutscene. Play / Restart checks your real startup and entry actions. Use all three for their intended jobs.

15. Make a shot list before adding every block

Write down the purpose of each part of the scene before filling the timeline. A short shot list prevents a common problem: lots of camera movement with no clear reason to look anywhere.

Time	Picture / action	Sound / speech
0-2	Workshop wide view; title appears and disappears.	A short bell accent.
2-4.4	Camera moves closer while Ivo approaches Elias; they face each other.	Leave space for the approach.
4.4-8.9	Elias close-up.	Every afternoon, that bell calls the town home.
8.9-10.8	Ivo reverse angle.	But not today.
10.8-14.7	Both characters in frame.	No chime. Just that uneven ticking.
14.7-18	Ivo acknowledges; pull back; mark completion.	Finish the line and return to gameplay.

This is a practical first scene, not a fixed cinematic formula. Use fewer shots if the exchange reads better that way. A clear, still shot is often more useful than an orbit through the wall.

> Appendix A lists the exact 18 events and their action data. Build the example in stages, then use that reference to check the complete schedule.

16. Prepare the room and actor marks

Use a walkable room with enough space for the planned views. The example floor is about 12 units wide and 9 units deep. Its navigation bounds cover the acting area, and the bench sits behind the conversation rather than across the walking route.

Item	Example setup
Ivo / ada starting mark	[-2.3, 0, 1.3]
Ivo destination	[-0.5, 0, 1.3]
Elias / elias mark	[1.2, 0, 0.2]
Floor / walk height	Y = 0
Wide camera position	[0, 4.5, 9]
Wide camera look target	[0, 1, -0.3]



Figure 7. Ivo approaches through open floor space while the camera moves closer.

First test that the player can walk to the destination during ordinary gameplay. A timed movement event does not fix a blocked route, a collision radius that is too large or navigation that ends before the destination.

> All XYZ examples use the room’s 3D world coordinates. Y is height in this workshop. Adapt the values to your scene scale instead of assuming every imported model has the same size.

17. Author named cameras in the room

1. In Scene Designer, add a Camera to the room and select it in the scene hierarchy.
2. Give it a clear name and ID. Edit Position & View, or frame the editor view and use Copy View to Camera.
3. Check its target height, FOV, clipping and any room camera bounds.
4. Apply the camera changes. Repeat for the close-ups and two-shot.
5. Keep wide as the room’s Default Camera for normal entry.

Camera ID	Position	Look target / FOV
wide	[0, 4.5, 9]	[0, 1, -0.3] / 42
elias_close	[-0.4, 1.65, 2.8]	[1.2, 1.35, 0.2] / 32
ivo_close	[2.1, 1.7, 1.8]	[-0.5, 1.3, 1.3] / 32
two_shot	[3.3, 2.7, 6.8]	[0.35, 1, 0.7] / 40

The close-up values assume Ivo has reached his destination. A camera aimed at that mark will not magically keep an actor framed if the actor is still walking elsewhere.

> A camera ID is local to the room. A ChangeCamera event needs that camera in the current room, not merely a similarly named camera in another scene.

18. Cut to a named shot or set a raw pose

Action	Use
SetCamera	A complete position, target and FOV authored directly on the event.
ChangeCamera	An existing camera ID in the current room.
ActivateCamera	An authored camera with an explicit cut/blend choice and optional advanced behavior.

```
{
  "type":"ChangeCamera",
  "cameraId":"elias_close"
}
```

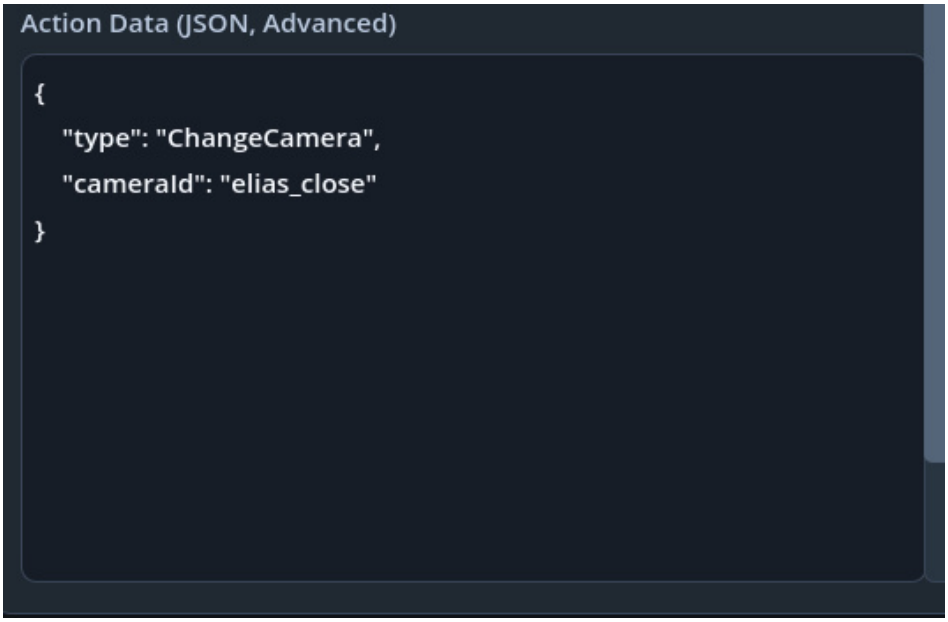

Figure 8. A named shot keeps its camera framing in Scene Designer while the timeline stores when to use it.

Place Elias’s cut at 4.4 seconds, just before his speech at 4.5. Put Ivo’s reverse at 8.9 and the two-shot at 10.8. Small gaps make the picture change easy to see before the next sentence begins.

Ordinary cameras switch immediately. When advanced behavior is involved, ChangeCamera can use a camera’s default blend. Use ActivateCamera with duration 0 when you need an explicit cut.

19. Move the camera over time

```
{
  "type":"CameraMove",
  "position":[0, 3.4, 7],
  "target":[0, 1, -0.2],
  "fov":42,
  "duration":2
}
```

1. Add CameraMove at 2.0 seconds on the Camera track.
2. Set the block Duration to 2.0 seconds and use the destination pose above.
3. Apply the event. The editor synchronizes the action duration with the block.
4. Test from the establishing shot and watch the whole move.

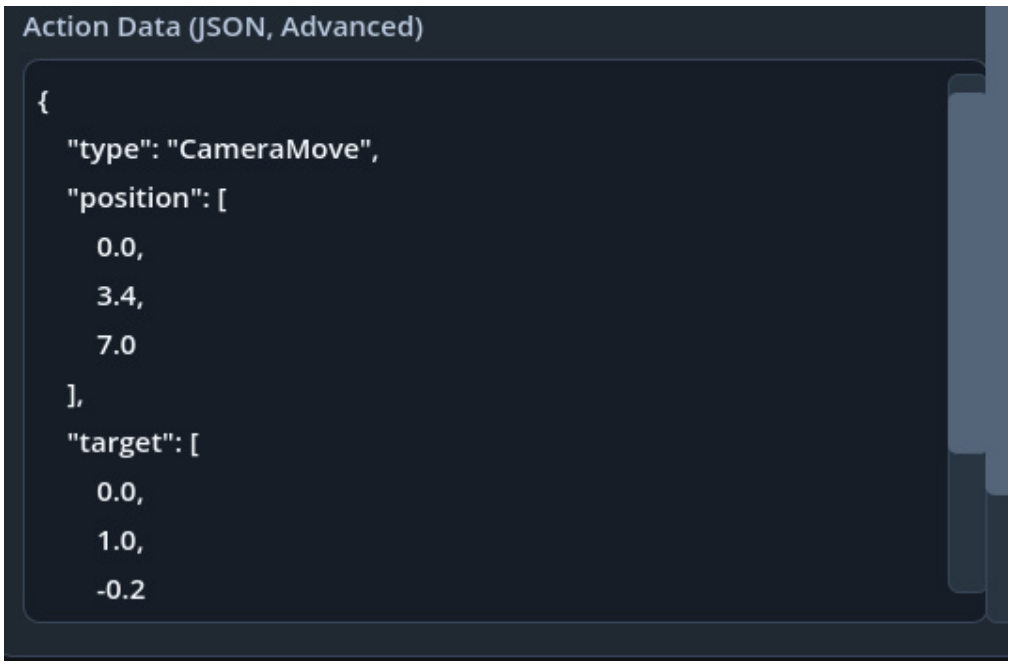

Figure 9. The destination is in Action Data. The event Duration controls how long the move takes.

The move starts from the current live camera pose and eases toward the destination. Both position and look target can move, and FOV can change as well. A changing FOV alters the lens-like view; it is different from physically moving closer.

Finish the move before the next unrelated camera cut. Two overlapping CameraMove events can keep writing to the same camera. A later cut alone is not a reliable substitute for ending the earlier move.

20. Blend between authored cameras

Activate / Blend to Camera exposes a camera selector and blend controls above the JSON box. It is a convenient way to reuse a carefully framed shot.



Figure 10. A timeline blend can run alongside movement and speech. Set Wait for Blend off for this scheduled use.

Blend Seconds	Result
0	Immediate cut.
Positive number	Blend for that many seconds.
-1	Use the destination camera’s default blend duration.

Use Wait for Blend in a sequential script or action list when the following step should wait. In the timeline, turning it on does not postpone unrelated scheduled events. Set it off and place the other events at their intended times.

The camera’s advanced settings supply its blend curve, such as sine or cubic. Test the start and finish against the actors’ marks; a smooth transition can still end on a badly framed shot.

> Manual SetCamera, CameraMove and orbit work can take camera ownership away from advanced tracking. ActivateCamera is the deliberate way to return to an authored advanced camera.

21. Follow a moving actor or frame a group

A moving actor is easier to frame with a tracking camera than with a guessed camera move. In Scene Designer, enable Advanced Camera on the chosen camera and set its behavior and targets.

Behavior	Useful for
fixed	A stationary authored view that can blend in and out.
follow	Maintaining an offset from a moving character.
group	Following several targets and optionally adjusting framing.
framed	Letting a target move inside a screen zone before following.
path	Following a route position related to the target’s location.



Figure 11. Advanced camera settings can be changed by a cutscene event without redefining the authored camera.

For Ivo, use targetId player and a target offset around [0, 1.3, 0]. For a group, add elias to targetIds. Check minimum/maximum framing limits and whether room bounds should constrain the shot.

> The path behavior is target-driven. It is not an automatically timed camera rail. Use CameraMove or a blend between fixed shots for an establishing move with a specific duration.

22. Change targets, priorities and orbit settings

Set Camera Targets changes who an advanced camera follows or looks at. Set Camera Priority can choose the highest-priority enabled advanced camera. Neither action creates a trigger area by itself; your timeline or scene event decides when to run it.

```
{
  "type":"CameraOrbit",
  "entityId":"elias",
  "speed":45,
  "distance":5,
  "height":2,
  "mode":"180_back",
  "clockwise":false,
  "repetitions":1
}
```

Orbit mode at 45 degrees/second	One repetition
360	A full turn: about 8 seconds.
180	A half turn: about 4 seconds.
180_back	Half a turn and return: about 8 seconds.

Orbit timing comes from speed, mode and repetitions. Applying the event recalculates its displayed duration from those settings. Merely stretching the bar is not the way to change the orbit’s angular speed.

Check the entire orbit for walls, backdrop edges and a view through the floor. Stop Camera Orbit ends orbit control deliberately; finishing a cutscene also clears its active orbit.

23. Move a character to a mark

```
{
  "type":"WalkTo",
  "entityId":"player",
  "position":[-0.5, 0, 1.3],
  "speed":1.2
}
```

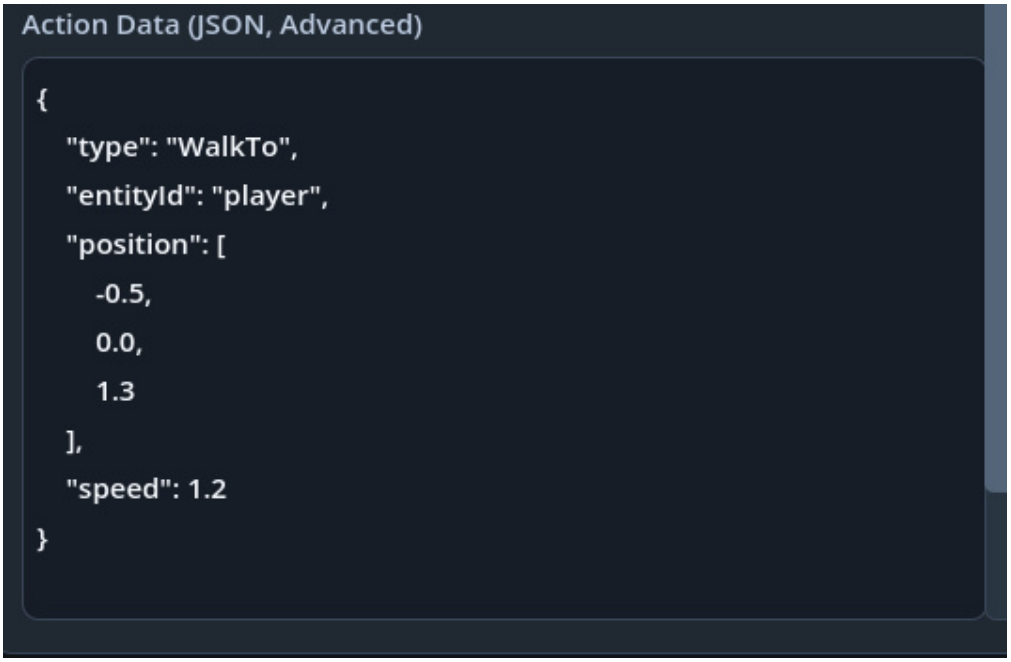

Figure 12. The movement target and speed belong to the action. The block length is your scheduling space.

At 2.0 seconds, Ivo begins walking toward the mark. The example reserves about 1.8 seconds before the facing actions. This includes a little room for arrival rather than treating distance divided by speed as a perfect guarantee.

WalkTo starts movement and lets the timeline continue. Bar duration does not wait for arrival and does not stop the actor at the bar’s end. Navigation, collision and the distance to travel all affect the real result.

A positive speed changes the actor’s walking speed. If it is a temporary cinematic speed, restore the normal speed in your cleanup. For a sequence that truly must await arrival, use a sequential movement script rather than another guessed timestamp.

24. Turn actors toward each other

```
{
  "type":"FaceCharacter",
  "characterId":"player",
  "targetCharacterId":"elias"
}
```

Place Ivo’s facing event at 4.2 seconds and an opposite event for Elias at 4.3. For Elias, set characterId to elias and targetCharacterId to player.

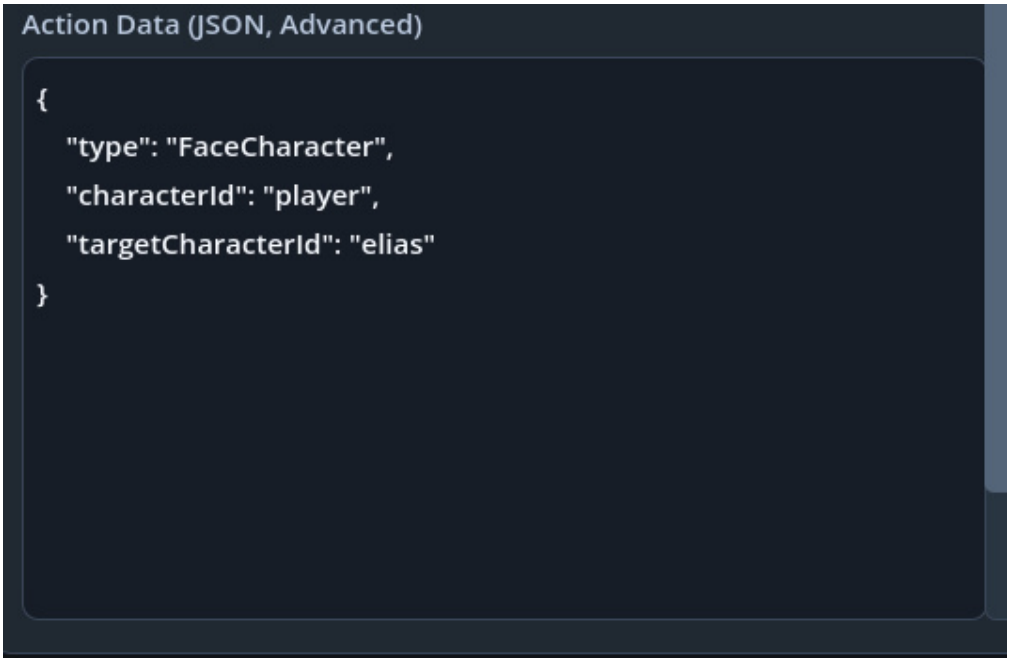

Figure 13. Facing is a separate action from walking. The character and target IDs must refer to the intended people.

Action	Target
FaceCharacter	Another character.
FaceObject	An object in the room.
FaceCoordinates	An explicit XYZ point.

Use FaceCoordinates for a mark with no visible object, such as a sound coming from a doorway. If the model faces backward despite a sensible target, inspect the character’s model-facing offset rather than reversing every event by guesswork.

Keep the walking route and facing moment separate while learning. A facing action dispatched before movement ends can be followed by the actor turning again to continue walking.

25. Use animation states and imported clips

Animation states give your project readable names such as idle, walk, nod and explain. They map to the real clips in the assigned model. Verify those mappings in Character Designer before scheduling gestures.

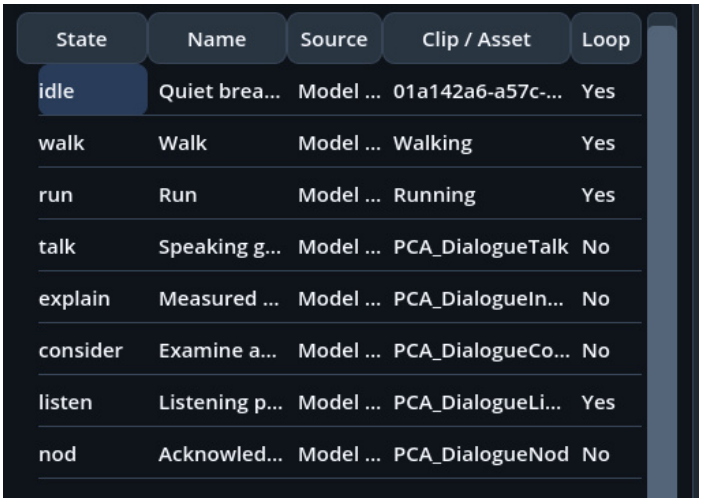

Figure 14. The bundled characters list mapped states. Select a state to inspect its exact clip.

Action / value	Meaning
SetCharacterState: stateId nod	Requests the character’s mapped nod state.
PlayAnimation: clip nod	Can use the mapped state where available; imported clip names must match real content.
Block Duration 0.5	A place in the schedule; it does not retime or trim the source model clip.

```
{
  "type":"SetCharacterState",
  "entityId":"player",
  "stateId":"nod"
}
```

Place the nod at 14.7 seconds. Let the earlier speech finish first. Automatic talk animation and a gesture on the same actor can otherwise replace one another.

If the actor slides or changes position unexpectedly, inspect root motion, model alignment and the state mapping. A timeline cannot supply an animation the model does not contain.

26. Schedule a timed spoken line

1. Add Say at 4.5 seconds on Dialogue.
2. Set Duration to 4.2 seconds, Label to Every afternoon... and speakerId to elias.
3. Enter the exact sentence and choose the matching voice recording when available.
4. Apply the event. The Say timeout is synchronized to 4200 milliseconds.
5. Place the next shot and reply after this line has enough space.

```
{
  "type":"Say",
  "speakerId":"elias",
  "text":"Every afternoon, that bell calls the town home.",
  "timeout":4200,
  "autoAdvance":true,
  "voiceAssetId":"voice_elias_01"
}
```

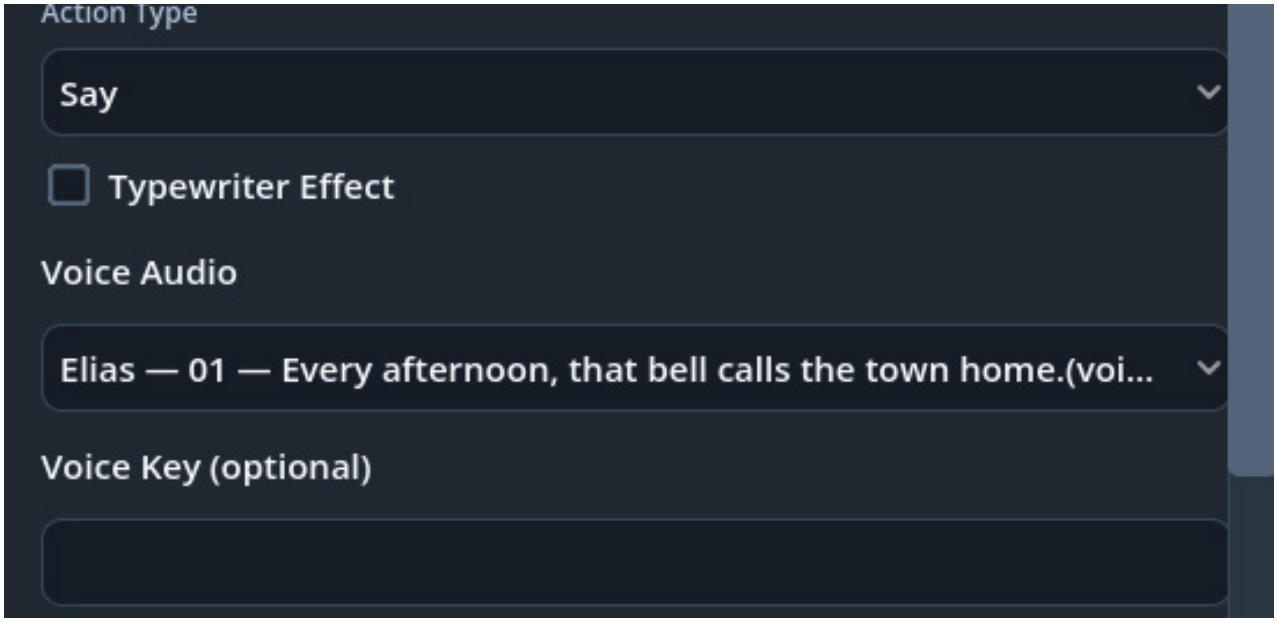

Figure 15. Timed speech has voice and typewriter controls as well as its action data.

Use the block Duration in seconds. The editor writes the speech timeout in milliseconds. Do not casually replace 4200 with 4.2 and then forget which field you are editing.

27. Add the recording and mouth timing

Import the voice recording as an audio asset, preview it and assign its ID to the matching Say event. Keep the displayed words consistent with the performance. The first example recording is roughly four seconds long.

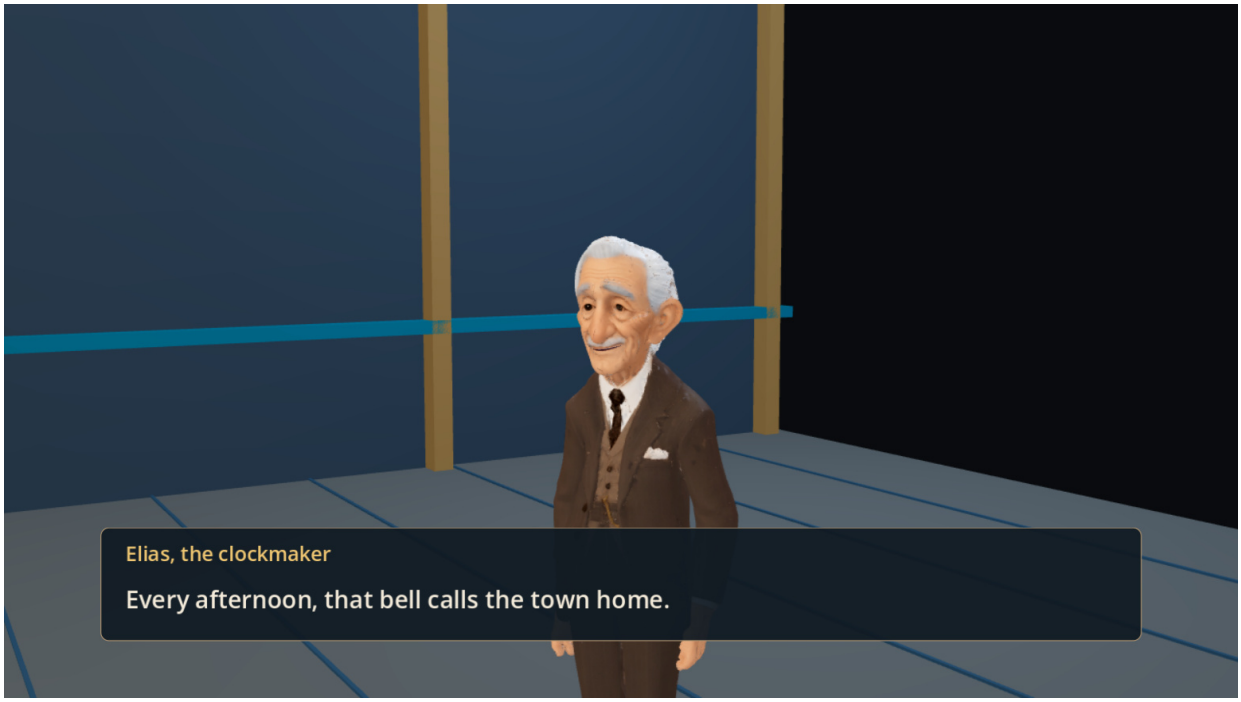

Figure 16. Elias’s close-up uses the real recording and a prepared mouth track on the scene character.

1. In the audio asset’s Lip Sync tool, generate, import or correct its recording-specific cue track and save it.
2. Enable lip sync on the character and choose Scene Character, or Both if a portrait also needs it.
3. Map the mouth targets that actually exist on the model.
4. Test the spoken line at its real camera distance.

A voice asset provides the sound; its cue track provides timing; the character mapping identifies the mouth shapes or bones. All three links need to be correct. A text-driven fallback is an approximation, not a replacement for a matching prepared track.

> No voice asset means text dialogue, not an automatically created performance. PlaySound is for general sound effects; use Say or PlayVoice for speech and its mouth binding.

28. Leave enough room for speech and typewriter text

Line	Start	Block length
Elias: Every afternoon...	4.5	4.2 seconds
Ivo: But not today.	9.0	1.6 seconds
Elias: No chime...	10.9	3.6 seconds

Timed speech can wait for its reveal and recording, but the timeline’s other events keep their scheduled times. A longer recording or slower typewriter effect does not automatically move the next camera shot or spoken event later.

Turn the typewriter effect on only when it suits the scene. Choose Default, None or a short custom blip sound. With recorded speech, a silent reveal often reads more cleanly. The option controls display; it does not stretch the recording.

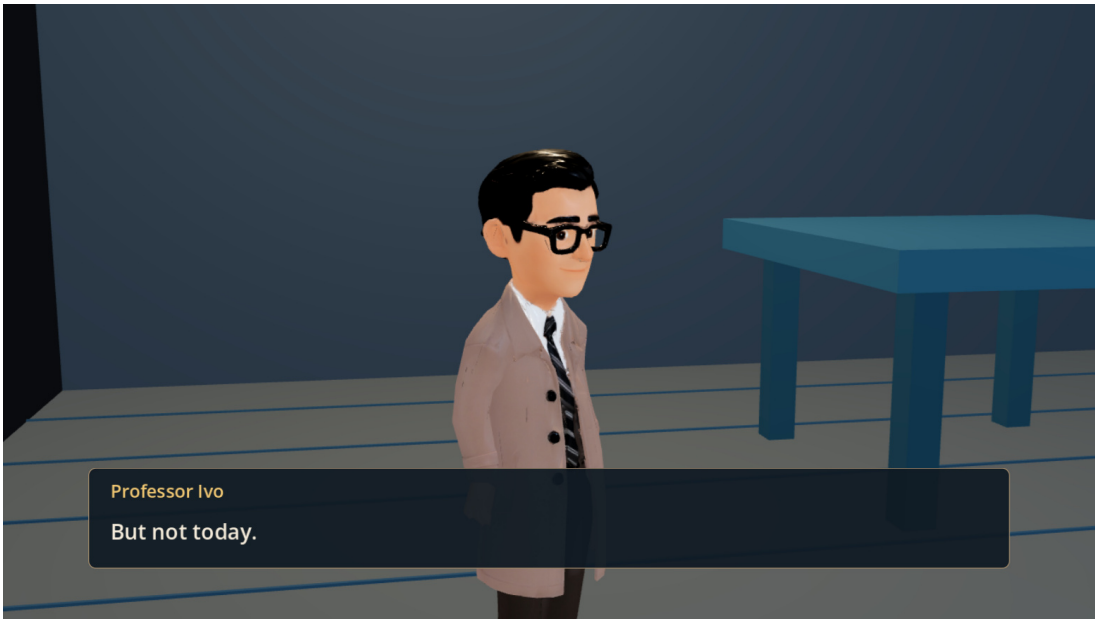

Figure 17. The reverse angle gives Ivo’s short reply its own readable moment.

After replacing audio, changing reveal speed or adding a translation, test the whole exchange. Keep a little breathing room between sentences instead of using the source clip’s exact final sample as the next event time.

29. Show a caption, name or changing value

ShowMessage is useful for narration or a short caption that does not belong to an ordinary character Line. Set its block Duration in seconds; applying the event writes the action’s duration in milliseconds.

```
{
  "type":"ShowMessage",
  "text":"Later that afternoon...",
  "duration":2500,
  "blockInput":false
}
```

Text token	Displayed value
@PlayerName	The active player’s display name.
@SceneName	The current scene name.
@Score	The score variable.
$$variable_id	The value of a project variable.

For example, Welcome back, @PlayerName. follows the active player’s name. Tokens supply text; they do not change a variable. An unknown variable token remains visible so you can spot the missing definition or spelling error.

The cutscene’s input-lock option already controls ordinary gameplay input. A caption’s blockInput field is a separate local presentation decision; do not scatter extra locks through every caption unless you need them.

> Caption appearance belongs to the relevant message/narrator interface. A title composed of several carefully positioned elements is usually better as an authored GUI.

30. Put sounds on named channels

```
{
  "type":"PlaySound",
  "assetId":"chime",
  "channel":"intro_bell"
}
```

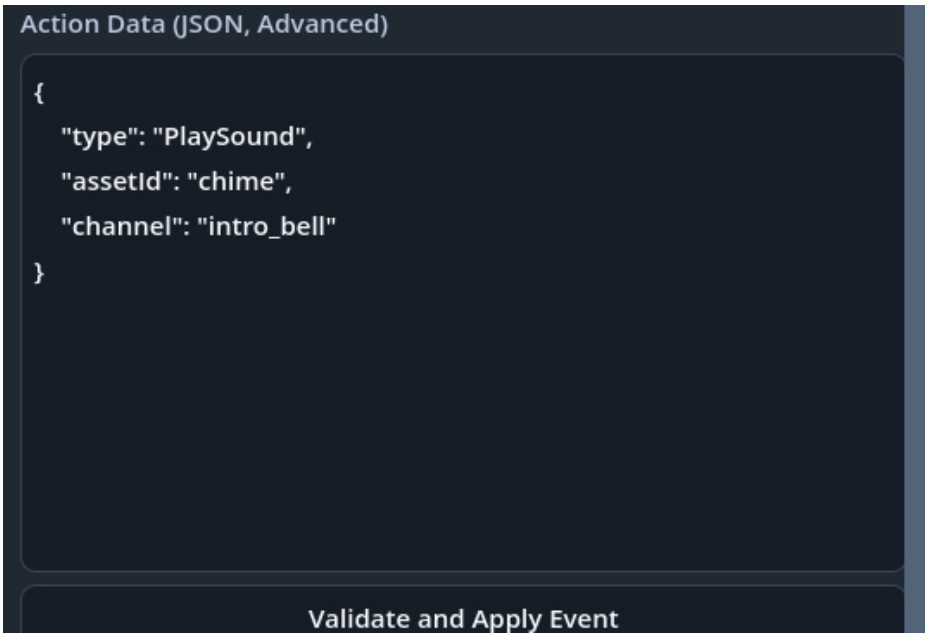

Figure 18. A named channel gives a later StopSound event an exact sound to stop.

Add the bell accent at 0.1 seconds and move it to Audio. PlaySound may initially land on another authoring track; the chosen lane does not change how it plays.

```
{
  "type":"StopSound",
  "channel":"intro_bell"
}
```

A second PlaySound on the same channel replaces the previous sound on that channel. Use different channel IDs for sounds that should overlap. If no channel is supplied, the asset ID is used as its channel reference.

The bar’s right edge does not stop the sound. Let a short recording end naturally, or schedule StopSound. A loop or long sound also needs cleanup when the player skips before its stop event.

31. Keep voice playback separate from sound effects

Use Say when a character’s text, recording and mouth timing belong together. Use PlayVoice when you need speech playback separately from a Say event, such as a voiceover or a specially staged caption.

```
{
  "type":"PlayVoice",
  "voiceAssetId":"voice_elias_01",
  "voiceKey":"",
  "text":"Every afternoon, that bell calls the town home.",
  "wait":false
}
```

Action	Use
SetVoiceSpeaker	Selects the default character for separate voice playback / supported captions. An empty character ID clears it.
PauseVoice / ResumeVoice	Controls the current speech recording.
SeekVoice	Changes voice playback position.
StopVoice	Stops speech playback; StopSound addresses sound-effect channels instead.

Wait for Voice is meaningful inside a sequential action flow. It does not retime the other blocks on a running cutscene timeline. The next scheduled Say can replace a voice that has not finished.

> Skipping stops the current voice through the cutscene runner. Ordinary completion does not mean every separately started long recording was deliberately stopped. Include that decision in your cleanup.

32. Build a readable subtitle interface

1. In GUI Designer, create a GUI named Cinematic Subtitles with ID cinematic_speech.
2. Add a background Panel near the bottom of the game viewport.
3. Add a speaker Label whose text is $$_dialogue_speaker.
4. Add a larger text Label with Group ID @talkdialogue.
5. Use trigger activation and start hidden. Begin with Cut transitions.
6. Assign it in Global Settings → System GUIs → Speech Bubble / Character Say.

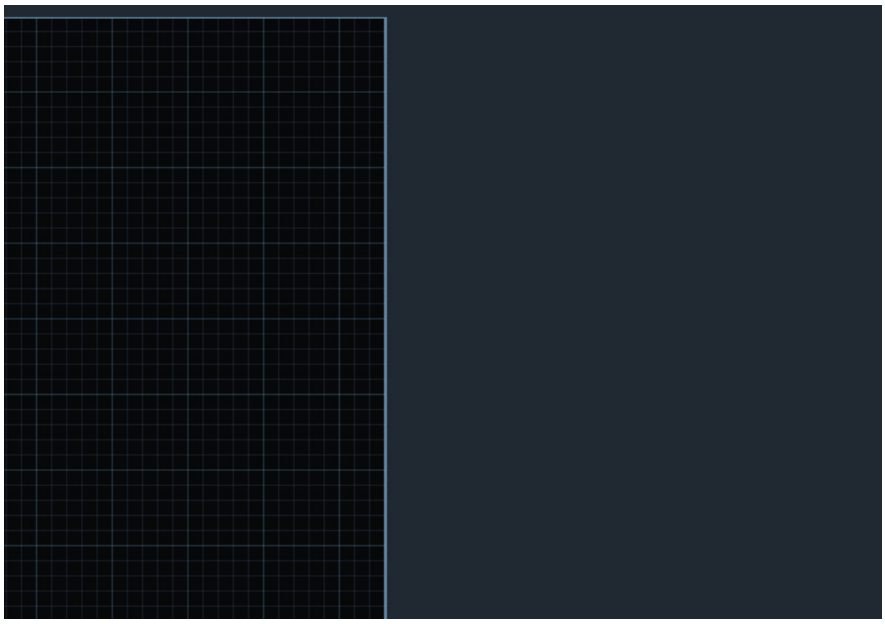

Figure 19. The subtitle Label receives speech through its Group ID. Its ordinary element ID can be speech_text.

Timed Say events outside a dialogue graph use the Speech Bubble / Character Say role, unless the active player has a personal dialogue GUI override. Dialogue graph Lines use the separate Dialogue role. Check the active player’s override if your global assignment seems ignored.

Use large text and a panel with enough contrast. The sample has no portrait slots so the room characters remain the focus. Optional portrait images use @playerpic and the case-sensitive @NPCpic bindings.

33. Make the opening title a GUI

Create title_card in GUI Designer. The sample uses a dark translucent Panel, a large centered title and a smaller subtitle. Its authored position is near the top of the viewport so it does not cover the characters.

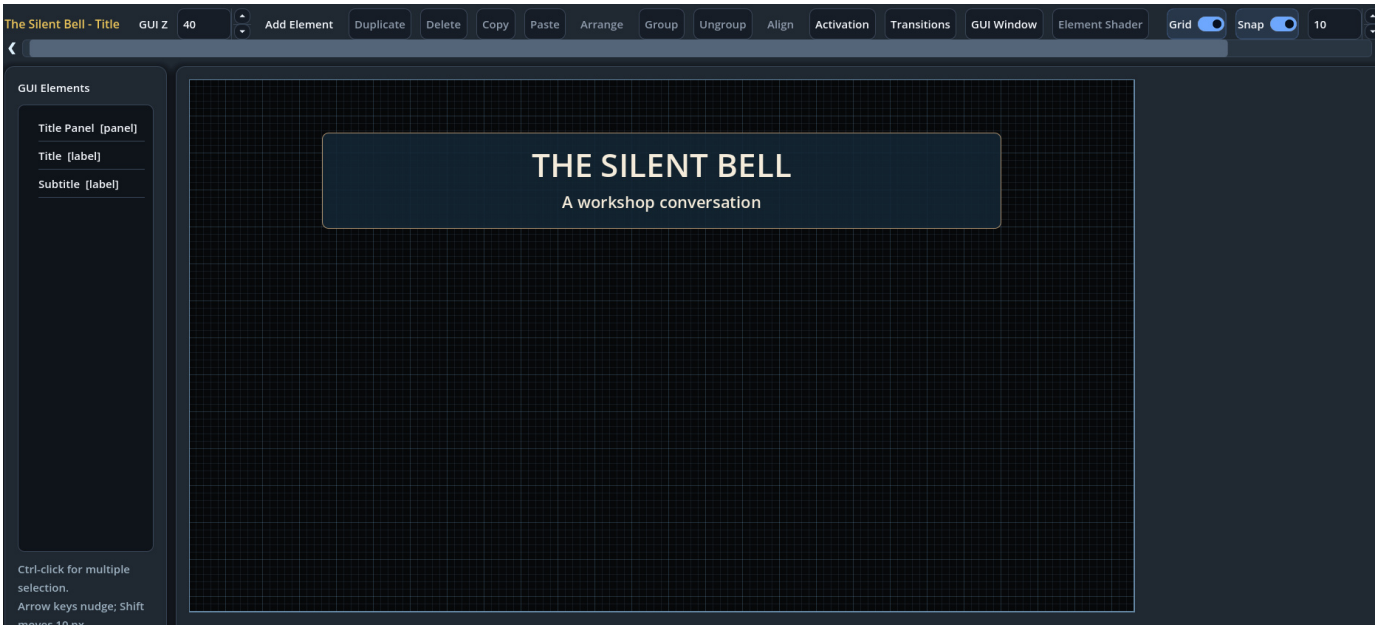

Figure 20. The title’s composition is edited in GUI Designer, independently of the timeline.

Time	Event action
0.2	ShowGUI, guild title_card.
1.8	HideGUI, guild title_card.

Start with Cut transitions to establish the timing. If you add a fade in GUI Designer later, allow time for it in the sequence. The cutscene event’s own 0.1-second block does not define the GUI fade duration.

A GUI displayed by ShowGUI is not automatically tied to the lifetime of that event block. Schedule the hide explicitly and hide it again in the shared cleanup for skip.

> Hide HUD hides the gameplay HUD. It does not mean “hide every authored GUI,” including the title and skip controls you just opened.

34. Understand automatic restoration

State	Cutscene runner’s finish behavior
Saved camera	Restored when Restore Camera is on, for completion and skip.
Gameplay HUD visibility	Restored to its saved visibility, rather than always forced on.
Mouse mode / pointer	Restored to the saved mouse mode.
Cutscene input lock	Released. Other independent game locks can still exist.
Tracked CameraMove tweens / camera shake	Stopped; active orbit is also cleared.
Current voice on skip	Stopped.
Actor positions, animation and speed	Need your own intended ending or restoration.
Titles, extra GUI layers and sound channels	Need explicit cleanup when they should close.
Variables, quests and inventory	Keep their game-state changes; they are not rolled back.

Restore Camera is useful when a scene should return to whichever gameplay shot was active before the cutscene. Leaving it off is useful when the final shot should become the new view. Test the final view in either case.

> Automatic restoration is not an undo transaction for the entire room. Build a cleanup that establishes a sensible playable state on every exit.

35. Use restrained camera and screen shake

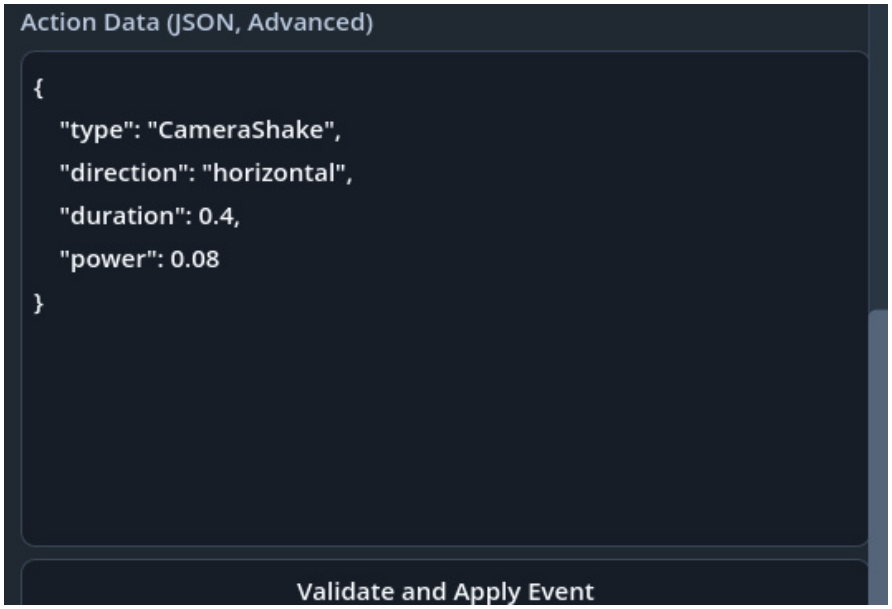

Figure 21. CameraShake has its own action duration and power. Applying a trim synchronizes this action’s duration.

```
{
  "type":"CameraShake",
  "direction":"horizontal",
  "duration":0.4,
  "power":0.08
}
```

A small shake can sell a bell strike or impact. Start with a low power and a short duration, then test the shot at the intended display size. A dramatic close-up needs less movement than a distant wide shot.

Action	Control
CameraShake	Small camera-view offsets; it ends after its configured duration or when the cutscene finishes.
ShakeScreen	A whole-screen effect with power expressed in pixels; StopScreenShake ends it.

Include a low-motion or no-shake route when it suits the game. Do not use shake to hide unclear animation or an awkward camera cut.

36. Change lighting, tint, glow and motion

The timeline can change room atmosphere or the state of an object. Examples include ToggleLight, SetSceneTint, SetGlow, SetMotion and their related toggle actions.

```
{
  "type":"SetSceneTint",
  "sceneId":"lab",
  "tintColor":"#b6c9e2",
  "strength":0.25
}
```



Figure 22. An atmosphere change is an action on the room, not a color painted onto the timeline track.

A Set action establishes an explicit value and is easier to make repeatable. A Toggle action reverses the current value, which may differ on a replay or a loaded game. Choose deliberately.

The event bar does not automatically remove the tint or restore the light. Add the opposite state when appropriate and repeat that restoration in cleanup. Keep story changes, such as a lamp being permanently repaired, separate from temporary staging.

A glowing material and a bloom-like rendering effect are different parts of the picture. Check the material, room effect settings and chosen renderer in the actual game preview.

37. Add screen effects without obscuring the scene

Set Retro Rendering Style and Zoom Blur expose dedicated parameter fields in the Cutscene Designer. They are useful for a dream, a sudden realization, a flashback or a momentary impact.



Figure 23. Zoom Blur provides strength, duration, center and clear-radius controls above its action data.

Effect	Useful decisions
Retro style	Choose none, psx, n64 or horror-fog; set strength and pixel size.
Zoom Blur	Keep the important subject near its center; leave a clear center radius.
Include GUI	Decide whether the effect should distort subtitles and overlays as well.
Stop Zoom Blur / style none	Provide a deliberate way to return to ordinary presentation.

An effect action’s own duration is distinct from a generic block width unless the editor synchronizes that action. For Zoom Blur, edit its Duration field and leave appropriate room in the schedule.

> The example introduction does not require these effects. Add one at a time in a separate test cutscene, then verify that the normal and skipped exits return to the intended screen style.

38. Call a script from an event

The Run Script action opens a script picker by name and ID. Its stored action type is RunVisualScript for compatibility, but the picker supports GDScript, PCA Script, Lua, BASIC and Visual records.

```
{
  "type":"RunVisualScript",
  "scriptId":"cinematic_helpers"
}
```

A normal Run Script event calls the script’s entry function. Use it for a small helper that changes a group of settings, stages a complicated pose or starts a background camera-follow task.

The timeline does not wait for that script to complete before dispatching later events. A helper that waits for the cutscene-finished signal can remain alive in the background, but it must filter the relevant cutscene and clean up deliberately.

Good helper job	Keep clear
Prepare / restore a group of temporary settings	Who owns the original values.
Control an advanced staged behavior	When it starts and stops.
Perform a one-time reward or game-state change	Its guard and replay behavior.

Do not use the outer introduction wrapper as a Run Script event inside the same introduction. It would start that cutscene again. The wrapper belongs to the scene, trigger, GUI or startup route that launches it.

39. Call exposed functions with parameters

For reusable helpers, expose a function through Functions as Nodes in the Script Designer. Define the actual code function name, its display name and parameters in the same order as the code, then choose Use Definitions.

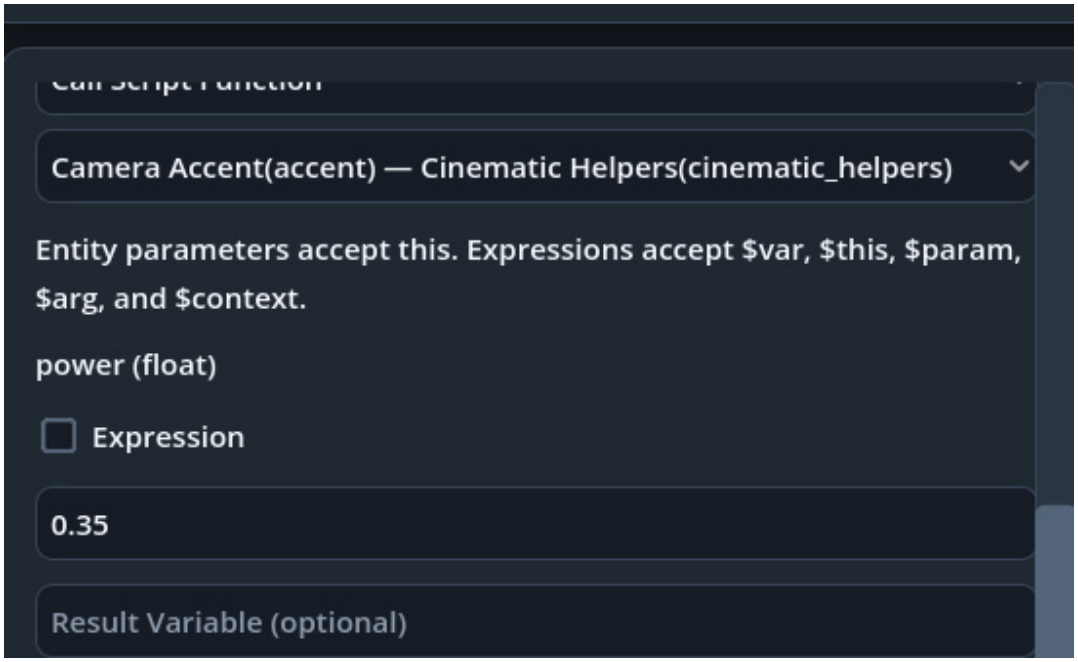

Figure 24. Call Script Function exposes the registered function’s editable parameters in the cutscene event.

```
func accent(power: float = 0.35) -> void:
    await ZoomBlur({
        "strength":power, "duration":0.6, "wait":false
    })
```

Put this function inside a script extending PCAScriptBase. Expose it as Camera Accent with one float parameter named power, default 0.35. Add Call Script Function to the cutscene, select Camera Accent and set its parameter.

Use Definitions saves the callable definition; it does not execute the effect. The event invokes it during playback.

> A caller-aware this value needs a real calling object in the captured context. A standalone Test Cutscene has no guaranteed object caller. Use explicit IDs in a reusable cinematic unless the caller is deliberately part of its design.

40. Finish the cutscene before opening choices

Say events are timed speech. A Dialogue Designer conversation can wait for player responses. Keep that distinction clear when combining the two tools.

```
extends PCAScriptBase

func run() -> void:
    await StartCutscene({"cutsceneId":"workshop_intro"})
    await StartDialogue({"dialogueId":"archive_chat"})
```

This sequential script waits for the cutscene to finish, then opens the branching conversation. Use the once-and-cleanup wrapper before starting dialogue when the introduction should also handle skip and replay state.

Approach	Result
StartDialogue event in a timeline	The conversation starts, but later scheduled timeline events still arrive.
Await cutscene, then StartDialogue	The conversation is a clear handoff after the cinematic.
Nested StartCutscene event	Uses the one cutscene runner; do not assume a parent cinematic will pause and resume like a nested dialogue.

For two cinematics in sequence, await them one after the other from an external action list or script. Do not build a chain by starting a second cutscene inside the first and expecting both timelines to retain independent control.

41. Use a movie or recorded playback deliberately

Start FMV plays an imported movie through the game’s movie surface. It is different from live camera/actor staging, and different from a Video element placed inside an authored GUI.

```
{
  "type":"StartFMV",
  "assetId":"demo_movie",
  "mode":"point",
  "endTime":2,
  "scriptId":""
}
```

Playback limit	Meaning
end	Use the normal end-of-recording completion route.
point	Pause at the requested movie time and use the point-completion route.
interrupt	Playback without the normal completion-script route.
Completion Script	An optional script for a deliberate end/point handoff.

Pause FMV, Resume FMV and Seek FMV control the movie’s own playback position. Their times are movie seconds, not the time ruler position in your cutscene. Stop FMV closes the movie playback surface.

A Start FMV block does not make the rest of the timeline wait for the movie. For a reliable cinematic-to-game transition, use an explicit completion route and test the imported movie on the intended target platform.

42. Props, attach points and other advanced events

Action family	Prepare first
Attach Object / Detach Object	The prop, its source attach point and the character’s target attach point.
Jump	A scene and character configured for the intended movement behavior.
Inventory / quests / score	Real item, quest, objective and variable IDs, with replay guards.
Start / Stop Gameplay Recording	A recording path and a deliberate capture period.
Play / Stop Gameplay Playback	A compatible saved gameplay recording and a safe place to apply its state.
Room change / character switch	The destination room, spawn and active player behavior.

A prop handoff is more than a timed attachment. Stage the actors, play the gesture, attach at the intended moment and establish who keeps the prop afterward. If a skip can occur before the attachment event, cleanup must still establish the correct result.

Gameplay recording/playback captures and reapplies runtime scene information. It is not a video export button for your timeline. Treat it as a separate system with its own playback and state consequences.

Additional shared actions can be called through scripts and action lists when they are not offered directly in Add Event. Use the normal action tools rather than inventing an unsupported type string.

> Keep a clear owner for every temporary prop, overlay and movement behavior. A timeline full of unrelated state changes becomes difficult to skip, replay and debug.

43. Choose the real launch point

Launch point	Useful for
Room Events → first entry	An introduction the first time the player visits a location.
Room entry every time	A repeatable arrival event; usually needs a story guard.
Trigger or object event	A discovery, trap, door interaction or scene reaction.
GUI event / visual script / action list	A deliberate cinematic command in an existing game flow.
Project Settings → startup options	A game-opening cutscene or startup script.

1. Select the room or object that should launch the introduction.
2. Open the relevant Room Events, Trigger Events or Verb Actions controls.
3. For the simplest raw test, add Start Cutscene and choose workshop_intro.
4. For the finished example, use Run Script and choose intro_once instead, so cleanup and the once-only guard surround playback.
5. Apply, save and test the real entry route with Play / Restart.

A project startup cutscene can run directly. To use the guarded wrapper at startup, choose the startup script route and select the wrapper. Confirm that the player’s room and required characters are loaded when it runs.

> Do not attach the same introduction to both startup and first entry without a deliberate guard. Test Cutscene alone does not prove that your real launch route fires exactly once.

44. Decide what “play once” means

A first-entry event and an intro_seen variable answer related but different questions. First entry belongs to the room’s visit history. The variable belongs to the story and can guard several launch routes.

```
if bool(Runtime.variables.get("intro_seen", false)):
    return
await StartCutscene({"cutsceneId":"workshop_intro"})
await SetVariable({"name":"intro_seen", "value":true})
```

Define intro_seen as a Boolean starting false. Place the guard in the script that launches the cutscene. The completed wrapper in Appendix B adds the required cleanup between playback and setting the flag.

Situation	Intended result
Fresh game	The introduction plays.
Player skips	It still counts as handled, if that is your story rule.
Player returns to the room	It does not replay.
A different button calls the same wrapper	The same guard applies.
New game / reset state	The initial false value permits a new introduction.

Do not put the only completion flag in the last scheduled event and assume it will run on skip. A skipped timeline does not fast-forward through its remaining actions.

45. Understand exactly what skip does

A skip request ends the running sequence when Skippable is enabled. It is not a request to execute all later events instantly. Any state change that has not been dispatched is still undone.

Skipped before this event	Possible leftover state
HideGUI title_card	The title can remain visible.
StopSound intro_bell	A long or looping sound can keep playing.
SetVariable intro_seen true	The next visit can replay the introduction.
A final character placement	The actor can remain at an intermediate position.
A reward / quest update	The story can lack a result you expected after the cinematic.

In the native test, skipping the raw example before its title-hide and completion events restored the camera, HUD and input but left the title visible and intro_seen false. That is why the finished launch route uses shared cleanup.

Decide whether each story consequence should happen even on skip. Usually the player should reach the same useful game state whether they watch a non-branching introduction or skip it. A deliberately interrupted failure scene may have different rules.

> Use Skip for an allowed skip request. An explicit End Cutscene/forced stop is a separate instruction and can terminate a cutscene even when its ordinary skip option is off.

46. Wrap playback in one cleanup route

The intro_once script starts the cutscene, waits for it to finish or be skipped, then establishes the intended playable state. That puts essential cleanup outside the part of the timeline that can be skipped.

Wrapper step	Why it is there
Check intro_seen	Avoid accidental replay through another launch route.
Save the current walking speed	Restore a temporary cinematic speed afterward.
Show skip controls when skippable	Provide a deliberate touch/mouse skip route.
Await StartCutscene	Run the remaining script after playback returns.
Hide title, skip and speech GUIs	Close authored overlays on both exits.
Stop the intro sound and voice	Avoid stray audio after the handoff.
Stop movement; restore speed; place Ivo	Establish the same final mark after a skip.
Return actors to idle; set intro_seen	Resume a clean game state and remember completion.

The complete script appears in Appendix B. It is intentionally specific to this workshop: ada and elias must be present in lab. Adapt the IDs and intended ending positions to your game.

> The raw timeline remains useful for editing. Ship the real entry route through the wrapper when you need its guard and cleanup.

47. Add a mouse or touch skip button

1. Create a GUI with ID skip_controls, hidden by default and activated by a trigger.
2. Place a readable Button in a screen corner. Give it a label such as Skip introduction.
3. Set its Click event to Run Script and select skip_intro.
4. Let the wrapper show the GUI before playback and hide it afterward.
5. Test clicking the real button while gameplay input is locked.

```
extends PCAScriptBase

func run() -> void:
    Runtime.cutscene_runner.skip()
```

This script requests a normal skip and respects Skippable. It is different from force-stopping the cutscene. If you make the sequence unskippable, do not present an apparently usable Skip button; the wrapper only shows it when the option is enabled.

Keep the GUI non-modal unless you deliberately need a modal layer. A cutscene input lock blocks ordinary world commands; it should not prevent your authored cinematic controls from doing their own job.

> A new cutscene does not automatically acquire the Clockmaker demo’s touch skip interface. That interface is authored content. Create and assign your own button or use an equivalent established project control.

48. Establish the same useful final state



Figure 25. After the guarded skip, the title is closed, the HUD is back and Ivo is at the intended playable mark.

Temporary change	Cleanup decision
Actor walk target	Stop movement before placing or handing control back.
Cinematic walk speed	Restore the saved speed.
Gesture or talking state	Return to the intended idle state.
Title, extra overlay, skip GUI	Hide explicitly.
Temporary sound / movie / screen effect	Stop or reset the specific system.
Permanent clue or quest result	Apply it through a guarded, deliberate state change.

Teleporting alone changes position; it does not necessarily clear an actor’s outstanding walking target. The example stops movement first. If your sequence changes rooms or replaces the controlled actor, adapt the cleanup to the new room instead of keeping a stale actor reference.

Choose an intended final state, rather than blindly restoring everything. Ivo should finish near Elias in this example, while the camera returns to its earlier gameplay view.

49. Use the right test for each job

Test	What it checks
Test Cutscene	The selected record in a restarted preview, with the current project data.
Test Scene	The room, actors, cameras and movement in an isolated scene workflow.
Play / Restart	The real startup, room events, trigger and wrapper route.
Packaged game	The final asset paths, renderer, controls and presentation outside the editor.

Apply the event before testing it. Save before the final pass. A timeline can be correct while the room’s entry action points to an old cutscene or calls the raw record instead of the wrapper.

When the introduction is already attached to a first-entry or startup action, watch for a duplicate start while using Test Cutscene. Use a controlled test copy or a dedicated launch route during authoring so you know which system started playback.

A once-only flag can make a repeat test appear to do nothing. Start fresh or deliberately reset the test state. Do not remove the guard from the shipped route simply because it did its job.

> Set up later shots with the same actor marks used during real playback. A close-up tested against a different starting pose can look broken even when the camera record itself is valid.

50. Run a completion and skip checklist

Test	Check
Watch from start to finish	Each shot, walk, line and title arrives at the intended time.
Skip while the title is open	Title and skip controls close; gameplay returns.
Skip during movement	Actor stops at the intended ending mark; speed is sensible.
Skip during a spoken line	Voice/mouth presentation and subtitles do not linger.
Skippable off	Ordinary skip request is ignored; input lock remains independent.
HUD initially hidden	Completion restores the saved hidden state rather than forcing it on.
Restore Camera on / off	The final view matches the selected policy.
Start through the real entry event	Runs once and uses the intended wrapper.
Restart after a handled introduction	Guard permits or blocks replay according to your design.
Load a save before / after	Story state, actor positions and entry history are consistent.

The guide’s native checks covered normal playback, recorded speech with a prepared scene mouth track, raw skip, wrapped cleanup, input/HUD/camera restoration, unskippable behavior and duration synchronization.

> Repeat the relevant checks after changing a voice recording, actor route, camera behavior, effect or cleanup script. Those changes can alter the handoff even when the timeline still looks tidy.

51. Read the bundled Clockmaker introduction

The bundled Clockmaker project is useful for studying a larger sequence. In this version, opening is A Minute Borrowed · directed opening: 49.8 seconds with 72 events. The exterior introduction is a separate 13-second sequence.

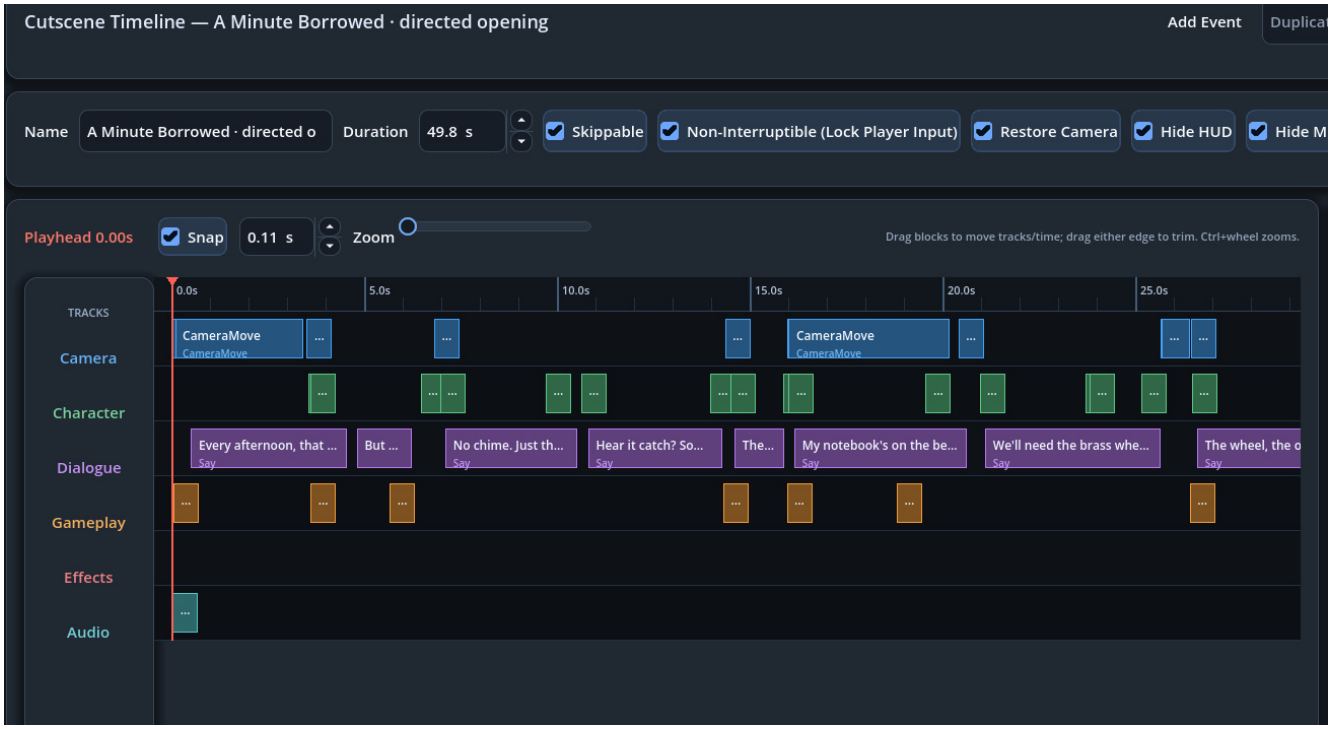

Figure 26. A portion of the real Clockmaker timeline. A long sequence still needs horizontal scrolling at the minimum zoom.

The opening combines mapped character states, named camera activations, timed speech, walks and script helpers. The current room entry routes call exterior_sequence or interior_sequence scripts, rather than relying only on a bare Start Cutscene action.

1. Open a fresh Clockmaker project or a copy you can inspect safely.
2. Select opening in Cutscenes and read the blocks by track.
3. Select an ActivateCamera event and identify the named camera in Scene Designer.
4. Select a Run Script event, then inspect the corresponding helper and its ending conditions.
5. Trace preparation and cleanup before copying any isolated event into another project.

> Older projects can contain a different Clockmaker introduction. Existing authored data is not automatically replaced by the latest template. Read the records in the project you actually opened.

52. Make the staging easy to follow

Give each shot a job. A wide shot explains where the people are. A closer view makes a reaction readable. A reverse angle lets the other person answer. A final wider view prepares the player to take control.

Practical choice	Benefit
Keep most of a conversation on one side of the actors’ eyeline	The viewer can follow who is looking at whom.
Cut shortly before a new line	The player sees who is about to speak.
Leave a quiet beat after important information	The clue has time to register.
Use a small gesture with a clear purpose	The actor reacts without constant animation noise.
Finish movement before precise facing or close-ups	The composition stays predictable.

Watch the exchange without reading the timeline labels. Then watch with sound muted and check that the staging still makes sense. Finally read the subtitles at the pace a new player would need, rather than the pace at which you remember the script.

> Simple staging is easier to maintain across different screen sizes, translated dialogue and modest hardware. Add complexity only when it improves the scene.

53. Fix timing and flow problems

Symptom	First checks
A later event starts too early	The timeline runs by time; it does not await the prior block.
Shortening total Duration does not end it earlier	A later event end can extend the effective runtime span.
A character keeps walking after the bar ends	WalkTo is a started movement, not a timed stop.
A cut is pulled back toward an earlier shot	An earlier CameraMove or tracking controller is still writing the camera.
A line is cut off by the next recording	Scheduled Say / PlayVoice events overlap.
Dragging a block does not trim its audio	The source asset is not edited by timeline trimming.
A title remains after skip	Its HideGUI event was never dispatched; add shared cleanup.
A second cutscene behaves unexpectedly	One runner is being asked to own two sequences. Chain outside with await.
Test starts at zero despite the playhead	The playhead is an editing position, not a playback seek point.

Reduce the sequence to one shot and one action when diagnosing a conflict. Add the next event only after you know which system controls the camera, actor or speech surface.

54. Fix missing cameras, actors and media

Symptom	First checks
Camera event appears to do nothing	The camera ID exists in the current room; apply the event.
Free camera ignored in a 2D Stage	The stage’s locked main-camera rules restrict arbitrary poses.
Actor fails to reach the mark	Navigation, collision, actual distance and entity ID.
Animation is missing or wrong	Assigned animated model, state mapping and exact clip name.
A raw pose looks through a wall	Position, target, FOV, clipping and room bounds.
No voice	Audio asset/key, voice language, packaged file and voice volume.
Voice but still mouth	Prepared recording track, character mapping and lip-sync target.
Subtitle GUI seems ignored	Speech Bubble role and the active player’s personal dialogue override.
Button cannot skip	Actual Click action, normal skip script and Skippable option.
An effect survives the cinematic	Its own stop/reset was omitted from cleanup.

A 2D Stage, a 2.5D scene and a free 3D room do not give every camera action the same freedom. Match the shot design to the scene’s camera model, rather than assuming a 3D orbit must work in a locked stage.

> Validate the project after editing IDs. An event that says “applied” can still refer to content that does not exist.

55. Check languages, controls and performance

Translate the original Say and caption text through the game translation tools. Keep text tokens intact and map the correct voice recording or voice key for each language. Generate or import a matching mouth track for each recording.

Check	Reason
Longest subtitle and title translation	Panel height and wrapping can change.
Longest localized voice	The fixed timeline may need a language-specific schedule or a sequential speech route.
Silent play / subtitles	Essential clues remain readable without audio.
Touch and controller use	The player needs a reachable skip or continuation route.
Reduced movement/effects option	Shake, orbit and blur can be replaced with a calmer presentation.
Target renderer and device	Lighting, material effects and performance can differ.

If performance is uneven, begin by reducing simultaneous effects, complex animated models and expensive views. Keep the intended event timing clear. A smooth editor timeline does not prove the scene renders smoothly on the player’s device.

> Do not assume a save resumes the exact active timeline time or movie position. Establish and test a checkpoint policy around the cutscene unless your game deliberately implements and verifies an in-progress resume system.

56. Finish the release pass

1. Apply every edited event and Save Project.
2. Use Sort and review the full timeline, including later events outside the visible area.
3. Use Auto Duration, then preserve any deliberate ending hold.
4. Run project validation and resolve missing targets, assets and invalid action data.
5. Test normal completion, early skip, mid-scene skip and the real launch route.
6. Check the final actor marks, camera, HUD, pointer, audio and temporary effects.
7. Check translated speech and the intended control devices.
8. Build the game and repeat the important routes outside the editor.

Keep stable IDs for cutscenes and their launch scripts. Record any save compatibility decisions when changing story flags or first-entry behavior. Remove temporary test launch routes from the game’s interface.

For a final pass, give the packaged game to a new reader and let them watch or skip without an explanation. They should reach the same clear starting situation for the next playable section.

> The appendices collect the full example timeline, the guarded launch/cleanup scripts and a compact timing reference. They are part of this PDF, with no separate practice project to download.

Appendix A. Complete workshop timeline

The next blocks form one complete cutscene record. Use them if you want the exact IDs and timings from the tutorial. They replace one cutscene record, not the whole project.

1. Create the room, actors, cameras, audio assets, title GUI and intro_seen variable described in the guide.
2. Create or select workshop_intro. Save a project copy first.
3. Open that cutscene record’s main Inspector → Raw JSON (Advanced).
4. Replace the record with all the following JSON blocks in order, including the opening and final braces.
5. Apply the record, inspect the timeline and save the project.
6. Test the raw timeline, then use intro_once for the finished guarded launch route.

Prerequisite	IDs
Room / player / NPC	lab / ada (player alias) / elias.
Named cameras	wide, elias_close, ivo_close, two_shot.
Recorded voices	voice_elias_01, voice_ivo_02, voice_elias_03.
Sound / title	chime / title_card.
Boolean flag	intro_seen, initially false.

The record does not create those prerequisites. Replace IDs when your project uses different records. The subtitle GUI is assigned separately through the speech system GUI role.

A1. Timeline record: lines 1-42

These are consecutive blocks of one JSON object. Copy code only, without page headings or captions.

```
{
  "id": "workshop_intro",
  "name": "The Silent Bell - Workshop Introduction",
  "duration": 18,
  "skippable": true,
  "lockPlayerInput": true,
  "restoreCamera": true,
  "hideHUD": true,
  "hideMousePointer": true,
  "events": [
    {
      "id": "establish",
      "time": 0,
      "duration": 0.1,
      "track": "camera",
      "label": "Workshop wide",
      "action": {
        "type": "SetCamera",
        "position": [0, 4.5, 9],
        "target": [0, 1, -0.3],
        "fov": 42
      }
    },
    {
      "id": "bell_sound",
      "time": 0.1,
      "duration": 0.1,
      "track": "audio",
      "label": "Bell accent",
      "action": {
        "type": "PlaySound",
        "assetId": "chime",
        "channel": "intro_bell"
      }
    },
    {
      "id": "title_on",
      "time": 0.2,
      "duration": 0.1,
      "track": "effects",
      "label": "Show title",
      "action": {
```

Continue directly with the next block.

A2. Timeline record: lines 43-84

These are consecutive blocks of one JSON object. Copy code only, without page headings or captions.

```
        "type": "ShowGUI",
        "guiId": "title_card"
      }
    },
    {
      "id": "title_off",
      "time": 1.8,
      "duration": 0.1,
      "track": "effects",
      "label": "Hide title",
      "action": {
        "type": "HideGUI",
        "guiId": "title_card"
      }
    },
    {
      "id": "push_in",
      "time": 2,
      "duration": 2,
      "track": "camera",
      "label": "Slow push in",
      "action": {
        "type": "CameraMove",
        "position": [0, 3.4, 7],
        "target": [0, 1, -0.2],
        "fov": 42,
        "duration": 2
      }
    },
    {
      "id": "ivo_walk",
      "time": 2,
      "duration": 1.8,
      "track": "character",
      "label": "Ivo approaches",
      "action": {
        "type": "WalkTo",
        "entityId": "player",
        "position": [-0.5, 0, 1.3],
        "speed": 1.2
      }
    },
```

Continue directly with the next block.

A3. Timeline record: lines 85-126

These are consecutive blocks of one JSON object. Copy code only, without page headings or captions.

```
    {
      "id": "ivo_faces",
      "time": 4.2,
      "duration": 0.1,
      "track": "character",
      "label": "Ivo faces Elias",
      "action": {
        "type": "FaceCharacter",
        "characterId": "player",
        "targetCharacterId": "elias"
      }
    },
    {
      "id": "elias_faces",
      "time": 4.3,
      "duration": 0.1,
      "track": "character",
      "label": "Elias faces Ivo",
      "action": {
        "type": "FaceCharacter",
        "characterId": "elias",
        "targetCharacterId": "player"
      }
    },
    {
      "id": "elias_shot",
      "time": 4.4,
      "duration": 0.1,
      "track": "camera",
      "label": "Elias close-up",
      "action": {
        "type": "ChangeCamera",
        "cameraId": "elias_close"
      }
    },
    {
      "id": "elias_line",
      "time": 4.5,
      "duration": 4.2,
      "track": "dialogue",
      "label": "Every afternoon...",
      "action": {
```

Continue directly with the next block.

A4. Timeline record: lines 127-168

These are consecutive blocks of one JSON object. Copy code only, without page headings or captions.

```
        "type": "Say",
        "speakerId": "elias",
        "text": "Every afternoon, that bell calls the town home.",
        "timeout": 4200,
        "autoAdvance": true,
        "voiceAssetId": "voice_elias_01"
      }
    },
    {
      "id": "ivo_shot",
      "time": 8.9,
      "duration": 0.1,
      "track": "camera",
      "label": "Ivo reverse",
      "action": {
        "type": "ChangeCamera",
        "cameraId": "ivo_close"
      }
    },
    {
      "id": "ivo_line",
      "time": 9,
      "duration": 1.6,
      "track": "dialogue",
      "label": "But not today.",
      "action": {
        "type": "Say",
        "speakerId": "player",
        "text": "But not today.",
        "timeout": 1600,
        "autoAdvance": true,
        "voiceAssetId": "voice_ivo_02"
      }
    },
    {
      "id": "two_shot",
      "time": 10.8,
      "duration": 0.1,
      "track": "camera",
      "label": "Both characters",
      "action": {
        "type": "ChangeCamera",
```

Continue directly with the next block.

A5. Timeline record: lines 169-210

These are consecutive blocks of one JSON object. Copy code only, without page headings or captions.

```
        "cameraId": "two_shot"
      }
    },
    {
      "id": "elias_reply",
      "time": 10.9,
      "duration": 3.6,
      "track": "dialogue",
      "label": "Uneven ticking",
      "action": {
        "type": "Say",
        "speakerId": "elias",
        "text": "No chime. Just that uneven ticking.",
        "timeout": 3600,
        "autoAdvance": true,
        "voiceAssetId": "voice_elias_03"
      }
    },
    {
      "id": "ivo_nod",
      "time": 14.7,
      "duration": 0.5,
      "track": "character",
      "label": "Ivo acknowledges",
      "action": {
        "type": "SetCharacterState",
        "entityId": "player",
        "stateId": "nod"
      }
    },
    {
      "id": "pull_back",
      "time": 15.2,
      "duration": 2,
      "track": "camera",
      "label": "Return wide",
      "action": {
        "type": "CameraMove",
        "position": [0, 4.5, 9],
        "target": [0, 1, -0.3],
        "fov": 42,
        "duration": 2
```

Continue directly with the next block.

A6. Timeline record: lines 211-237

These are consecutive blocks of one JSON object. Copy code only, without page headings or captions.

```
      }
    },
    {
      "id": "intro_complete",
      "time": 17.4,
      "duration": 0.1,
      "track": "gameplay",
      "label": "Mark intro seen",
      "action": {
        "type": "SetVariable",
        "name": "intro_seen",
        "value": true
      }
    },
    {
      "id": "bell_stop",
      "time": 17.6,
      "duration": 0.1,
      "track": "audio",
      "label": "Stop intro channel",
      "action": {
        "type": "StopSound",
        "channel": "intro_bell"
      }
    }
  ]
}
```

End of the complete timeline record. Apply the record, save and test.

Appendix B. The guarded launch script

Create an enabled project GDScript record with ID intro_once. Paste the next page’s complete source and save it. Leave it non-global with no automatic event subscriptions; the room/trigger/GUI launch action calls it explicitly.

Requirement	Why
Ivo is ada and exists in lab	The script stores that actor reference and walking speed.
Elias is elias	Cleanup returns the NPC to idle.
workshop_intro exists	The wrapper waits for this record.
title_card, skip_controls, cinematic_speech	These authored overlays have deliberate visibility cleanup.
intro_bell sound channel	Stops the introduction’s channel specifically.
intro_seen Boolean	Guards the introduction and records that it was handled.

The final player mark is [-0.5, 0, 1.3]. That is a story decision: Ivo should finish near Elias. The camera returns through the cutscene’s Restore Camera option.

This wrapper assumes the sequence stays in the workshop and keeps those actors alive. A room-changing cinematic needs a cleanup written for the destination room and its live actors.

> Keep this script outside the timeline it launches. Choose Run Script → intro_once on the real room, trigger or GUI event. Calling it from that same cutscene would start the cutscene again.

B1. Complete introduction wrapper

```
extends PCAScriptBase

func run() -> void:
    if bool(Runtime.variables.get("intro_seen", false)):
        return
    var actor: Node = Runtime.entity_nodes.get("ada")
    var normal_speed: float = float(actor.get("walk_speed"))
    var scene: Dictionary = Runtime.project.get_cutscene("workshop_intro")
    if bool(scene.get("skippable", true)):
        await ShowGUI({"guiId":"skip_controls"})
    await StartCutscene({"cutsceneId":"workshop_intro"})
    await HideGUI({"guiId":"title_card"})
    await HideGUI({"guiId":"skip_controls"})
    await HideGUI({"guiId":"cinematic_speech"})
    await StopSound({"channel":"intro_bell"})
    Runtime.stop_voice()
    actor.call("stop_moving")
    actor.set("walk_speed", normal_speed)
    await CharacterTeleport({
        "characterId":"ada", "sceneId":"lab",
        "position":[-0.5, 0.0, 1.3]
    })
    await PlayAnimation({"sceneId":"lab",
        "entityId":"ada", "clip":"idle"})
    await PlayAnimation({"sceneId":"lab",
        "entityId":"elias", "clip":"idle"})
    await SetVariable({"name":"intro_seen", "value":true})
```

Use your own actor IDs and final mark when adapting the example. The guard and cleanup run around the timeline, so an early skip does not bypass them.

B2. Skip button and reusable accent

skip_intro: normal skip request

```
extends PCAScriptBase

func run() -> void:
    Runtime.cutscene_runner.skip()
```

Assign skip_intro to the skip Button’s Click event. It respects the cutscene’s Skippable option.

cinematic_helpers: entry and exposed function

```
extends PCAScriptBase

func run() -> void:
    await accent(0.35)

func accent(power: float = 0.35) -> void:
    await ZoomBlur({
        "strength":power, "duration":0.6, "wait":false
    })
```

Run Script calls run(), which uses the default accent. To edit power on a cutscene event, expose accent as Camera Accent, with one float parameter named power and default 0.35. Choose Use Definitions, then select Call Script Function.

> Adding parameters to the callable definition does not rewrite the code function. Keep the definition’s names, types and order aligned with the actual function.

Appendix C. Timing and ownership reference

Item	Unit / ownership rule
Cutscene Duration	Seconds; runtime also considers the last event end.
Event Start Time / Duration	Seconds from cutscene start / displayed block span.
CameraMove	Block trim synchronizes its action duration in seconds.
CameraShake	Block trim synchronizes its action duration in seconds.
Say timeout	Written in milliseconds by applying the block duration.
ShowMessage duration	Written in milliseconds by applying the block duration.
CameraOrbit	Speed in degrees/second; width derived from mode and repetitions.
WalkTo	Speed and destination drive movement; bar width does not guarantee arrival.
PlayAnimation	Uses the mapped state or source clip; the bar is not a clip editor.
PlaySound / PlayVoice	Source playback is not cropped by the event bar.
ShowGUI	Visibility persists until hidden by its own flow or cleanup.
ZoomBlur	Use the effect’s own duration field and explicit reset when needed.
FMV Seek / Play Until	Movie playback seconds, independent of timeline position.
Wait in a called action/script	Waits that flow; does not postpone other scheduled events.

Choose ordinary readable durations for captions and speech, then apply the event. Recheck the stored action after trimming rather than assuming every action uses the same unit.

Appendix D. The complete event schedule

Start	ID / action	Duration
0	establish / SetCamera	0.1
0.1	bell_sound / PlaySound	0.1
0.2	title_on / ShowGUI	0.1
1.8	title_off / HideGUI	0.1
2	push_in / CameraMove	2
2	ivo_walk / WalkTo	1.8
4.2	ivo_faces / FaceCharacter	0.1
4.3	elias_faces / FaceCharacter	0.1
4.4	elias_shot / ChangeCamera	0.1
4.5	elias_line / Say	4.2
8.9	ivo_shot / ChangeCamera	0.1
9	ivo_line / Say	1.6
10.8	two_shot / ChangeCamera	0.1
10.9	elias_reply / Say	3.6
14.7	ivo_nod / SetCharacterState	0.5
15.2	pull_back / CameraMove	2
17.4	intro_complete / SetVariable	0.1
17.6	bell_stop / StopSound	0.1

The cutscene’s declared duration is 18 seconds. Events with a 0.1-second marker normally set or start something; their state does not automatically reverse at the marker’s right edge.

> The raw intro_complete event marks the normal ending. The guarded wrapper also sets intro_seen after its cleanup, which handles skipping before that final event.

Version and verification notes

Prepared against PCA 3D Adventure Studio 0.86.3.4 and its native cutscene editor/runtime. The screenshots show the actual editor and Play view. No engine source changes are needed for these instructions.

The working workshop example was exercised through its scheduled camera views, movement, recorded speech and final handoff. A prepared voice track was active on the scene character during the captured Elias line.

Checks also covered raw skip leaving unfired state changes undone, wrapper cleanup, the real GUI skip button, skippable-off behavior, restoration of camera/HUD/input, duration synchronization and an exposed function with a typed parameter.

The current Clockmaker project was inspected directly. Its 49.8-second directed opening and script-based room-entry routes differ from older template documentation. Existing projects may retain their older authored sequence.

The complete example JSON and GDScript are included as reading references in this PDF. No separate practice files are included. For a later engine release, compare control names, available action fields and the skip/cleanup behavior before reusing advanced recipes unchanged.