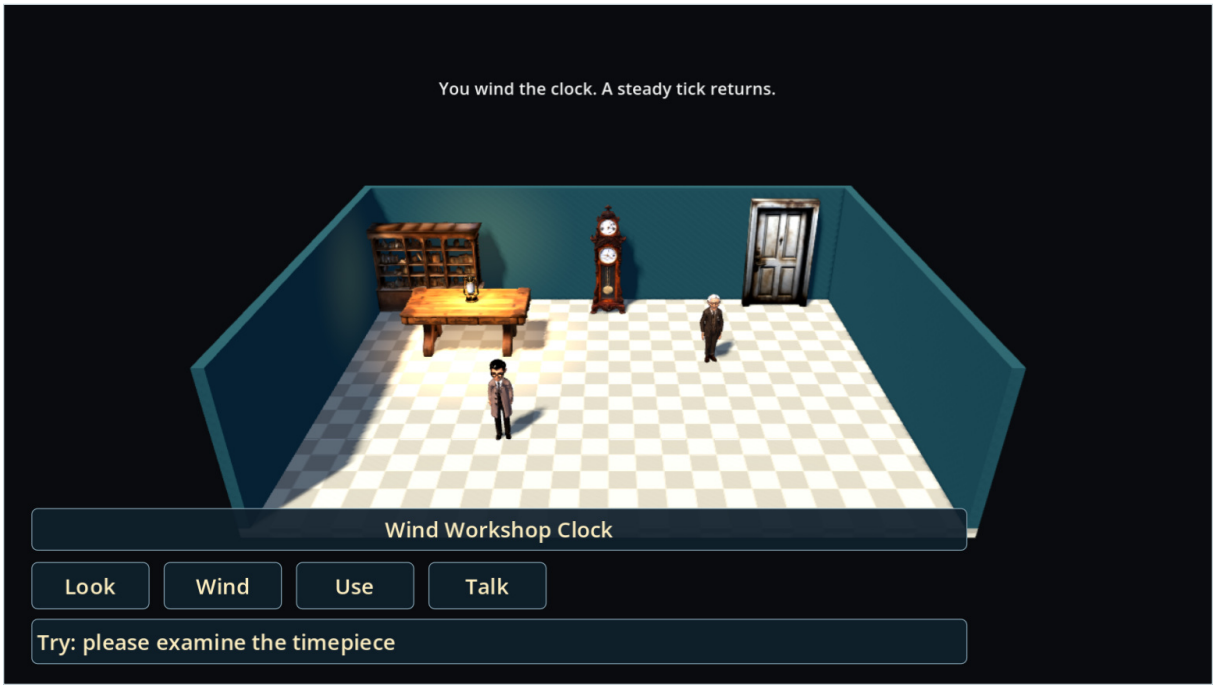
PCA 3D ADVENTURE STUDIO

Verbs & Mouse Cursors

Commands, pointers and the choices players make



Build a command and follow it into play

Learn the Verb Designer and Cursor Designer, give an object a response, and connect the same command to mouse input, a GUI button and typed text. Clear steps, actual editor screenshots and examples checked in the running game.

Version 0.86.3.4 - Windows, macOS and Linux

No practice project or extra downloads required.

How to use this guide

Start with one object and one new command. The Wind example gives you a small complete chain to follow. You can use your own scene and assets; the workshop names simply make the screenshots easier to read.

Chapters	What you will learn
1-6	Understand the command/target relationship, edit every verb field, create Wind, author a response and choose an interaction mode.
7-10	Create cursor records, position hotspots, set pointer size, understand priority and use object hover/press cursors.
11-14	Connect a GUI button and command label, try basic text parsing, add a command-entry box and set up a two-object command.
15-18	Use shared actions/scripts, test the complete chain and troubleshoot common mistakes.

The screenshots were captured from the running editor and game preview. Close-ups are cropped for readability. One desktop capture includes the actual mouse cursor. Existing studio artwork and the workshop assets provide the examples; no separate practice files are included.

The main subject is command and cursor authoring. The text-parser and GUI chapters cover the connections you need for this example, rather than every feature of those designers.

Apply Changes, Save Project, then test. Designer edits and runtime actions are different steps. Use a fresh game preview to check accepted authoring changes.

The guide follows version 0.86.3.4. Platform file pickers may differ, while the project fields and workflow are shared. Touch and gamepad presentation need their own device checks.

Contents

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

1. How the pieces fit together 4
2. Find the two designers and save your work 5
3. Understand every Verb Designer field 7
4. Create the Wind command 10
5. Give the clock its response 13
6. Choose how the player selects commands 15
7. Understand every Cursor Designer field 17
8. Set the click point and pointer size 19
9. Connect the global cursor settings 21
10. Give an object hover and pressed cursors 23
11. Add verb buttons and a command label to a GUI 25
12. Give the text parser the same vocabulary 28
13. Put a parser input box in the GUI 31
14. Use one object on another 33
15. Change verbs and cursors through actions or scripts 38
16. Check the whole chain in play 39
17. Fix common problems 42
18. A compact reference for your next project 43

1. How the pieces fit together

A player chooses **Look**, clicks a clock, and reads a description. That little exchange involves several parts of the game, but each part has a straightforward job.

Part	Its job in the example
Verb Designer	Defines Look as a command, gives it a display name and parser synonyms, and links a cursor.
Cursor Designer	Defines the pointer image and the point inside that image used for clicking.
Object's Verb Actions	Supplies what Look does to this particular clock.
GUI Designer	Provides a Look button, if you want one, and a label showing the current command.
Text Parser	Recognises a typed command such as "examine the clock" and routes it to the same verb and target.

The verb is shared across the adventure. The response belongs to the thing being used. **Look** can describe a clock, a door or a character without needing three separate Look verbs.

A cursor is the visual pointer. Giving a verb a hand-shaped cursor does not make the character pick something up. You still author that response on the object or inventory item.

What you will build

The example adds **Wind** to a small workshop adventure. Wind uses a hand cursor. The workshop clock has a Wind response that records a Boolean variable and displays a message. A GUI button selects Wind, and the parser recognises **wind up the clock** as the same command.

You will also look at a two-object command, **Use the lantern on the clock**, to understand the preposition controls in the Verb Designer.

Use a room and objects from your own project. The workshop screenshots provide names to follow; you do not need a practice project or a separate asset download.

> **Before you continue**
> Build one complete chain first: verb, cursor, target response, then a working test. Add more commands after that chain works.

2. Find the two designers and save your work

The workspace toolbar has separate icons for the Verb Designer and Cursor Designer. Hover over an icon to read its tooltip if you are unsure which one it opens. The cursor icon resembles a pointer; the verb icon opens **Verb Command Designer**.

You can also select an existing record through the Adventure Database. The designer's selector at the top lets you switch among records of that type.

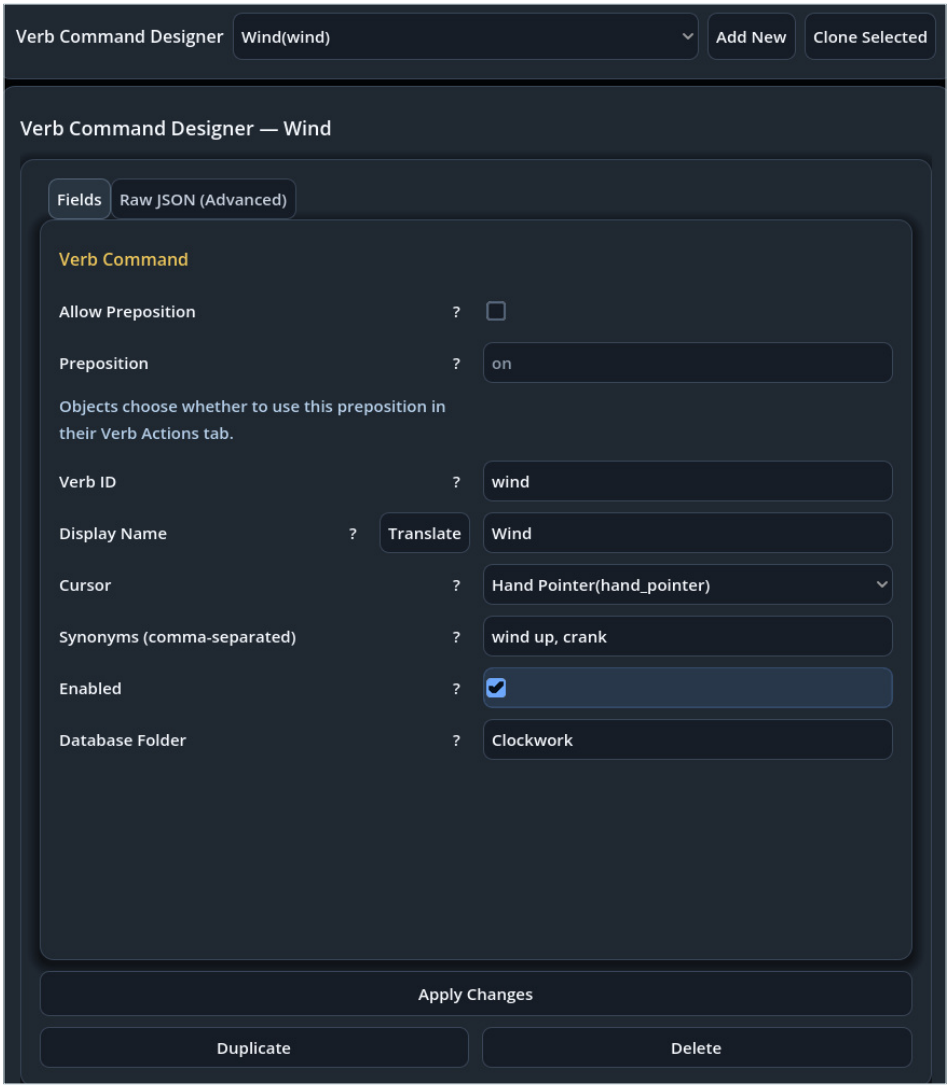


Figure 1. The Verb Command Designer has a record selector, Add New and Clone Selected above the selected verb's fields.

Read the controls in order

Control	What to do with it
Record selector	Choose the existing verb or cursor you want to edit. Labels show Name(ID).
Add New	Create a new record of this designer's type.
Clone Selected	Make another record from an existing one, then give the copy its own identity.
Fields / General	Use the ordinary editing controls. Verbs use a Fields tab; cursors use General.
Raw JSON (Advanced)	Inspect or deliberately edit the selected record as data. Use the normal fields while learning.
Apply Changes	Accept the edited fields into the open project.
Duplicate / Delete	Duplicate or remove the selected record. Check references before deleting something in use.
Save Project	Write the adventure, including accepted edits, to disk.

Apply, save, then restart a test

1. Change the fields in the correct designer.
2. Click **Apply Changes** beneath that form.
3. Click **Save Project** on the main toolbar.
4. Stop an old game preview and start a fresh test when checking authored changes.

A game already running in the preview has its own runtime state. Editing a verb record in the designer is not the same operation as issuing a runtime Change Verb action.

If you leave a form while it contains unapplied work, the editor may ask what to do with that draft. Apply the change you intend to keep before moving to the next designer. This avoids wondering later whether a field was saved or discarded.

Before deleting or renaming a record

Look for references from verbs to cursors, GUI events to verbs, object responses to verb IDs, and global cursor settings to the available verb list. Changing an ID can leave those references pointing to the old value.

Prefer changing a **Display Name** or **Name** when the goal is simply to improve the wording. Keep established IDs stable once your adventure uses them.

3. Understand every Verb Designer field

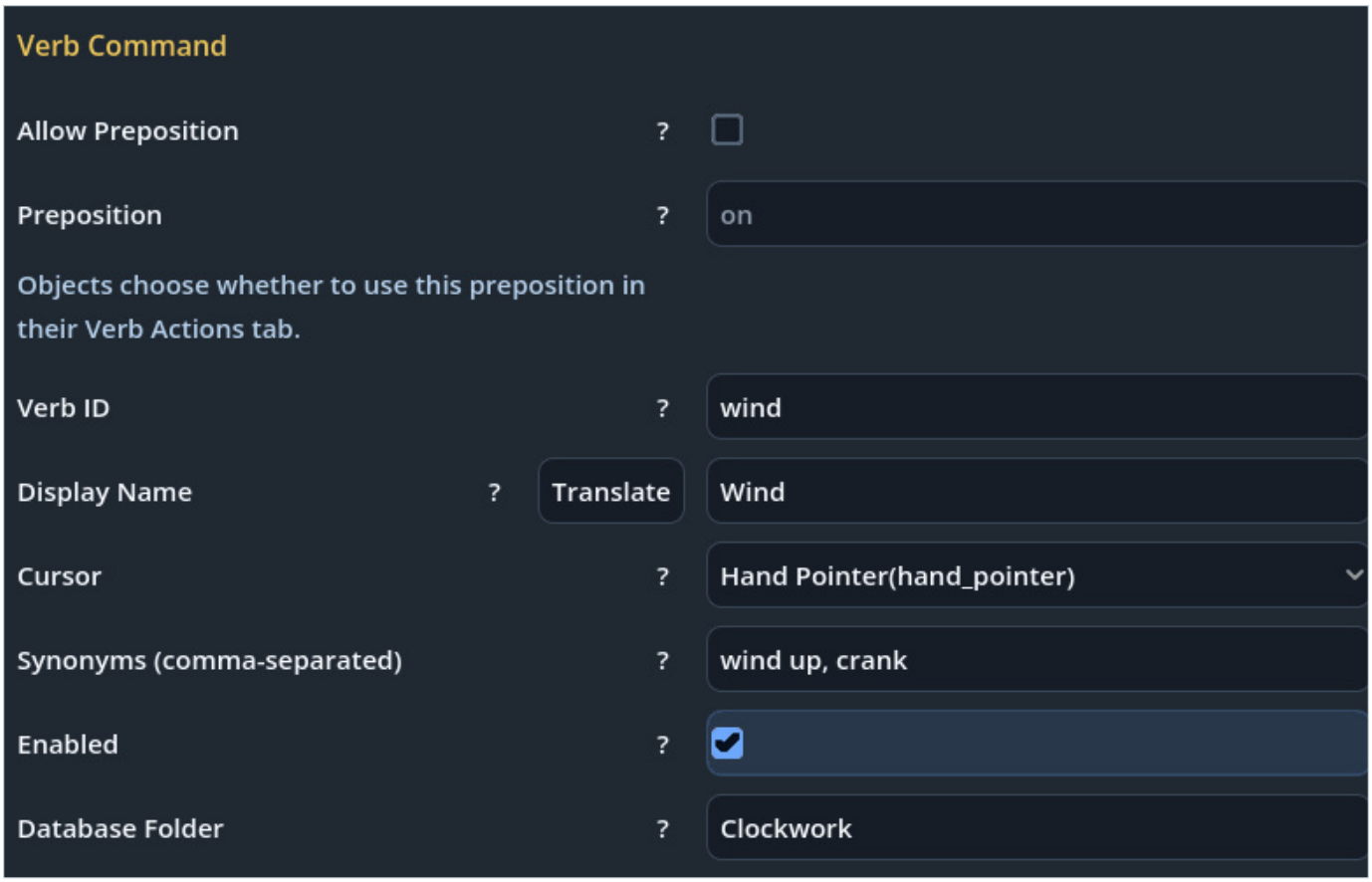


Figure 2. The completed Wind definition. It names the command, links Hand Pointer and provides two parser synonyms.

Verb ID and Display Name

Verb ID is the reference used by actions and saved project data. Use a short, unique value such as `wind`, `look` or `unlock`. The editor normalises a verb ID to lowercase and changes spaces to underscores when applying it.

Display Name is the wording players read, such as **Wind**, **Look** or **Pick Up**. For the familiar Take command, you can keep ID `take` and use **Pick Up** as its display name.

Avoid renaming a built-in ID just to alter its caption. Walk, Look, Use and inventory handling have particular runtime behaviour associated with their IDs.

Field	Wind example	Purpose
Verb ID	`wind`	Stable command reference.

Field	Wind example	Purpose
Display Name	Wind	Readable caption.
Cursor	Hand Pointer(hand_pointer)	Cursor record requested when this verb is selected.
Synonyms (comma-separated)	`wind up, crank`	Alternative words the parser can recognise.
Enabled	Checked	Include this authored verb in normal enabled-verb availability.
Database Folder	`Clockwork`	Organise the record in the project.
Allow Preposition	Unchecked	Keep this example as a one-target command.
Preposition	Empty	Used only when you configure a two-object command.

Cursor

The Cursor selector points to a **cursor record**, not directly to an image asset. That cursor record, in turn, chooses the image.

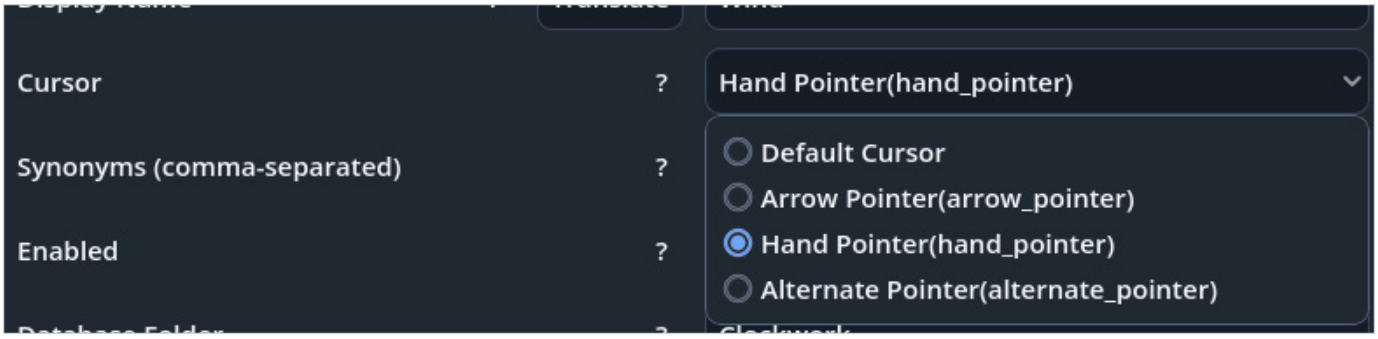


Figure 3. A verb selects a named cursor record. Several verbs can share the same cursor.

Choosing **Default Cursor** means the normal system pointer for that authored verb. It does not mean "find a cursor with the same name anyway". Assign a cursor explicitly when you want a custom pointer.

Synonyms

Enter alternatives separated by commas: **examine, inspect, observe** is a useful Look set. You do not need to include the verb's own ID and display name again; those are already aliases.

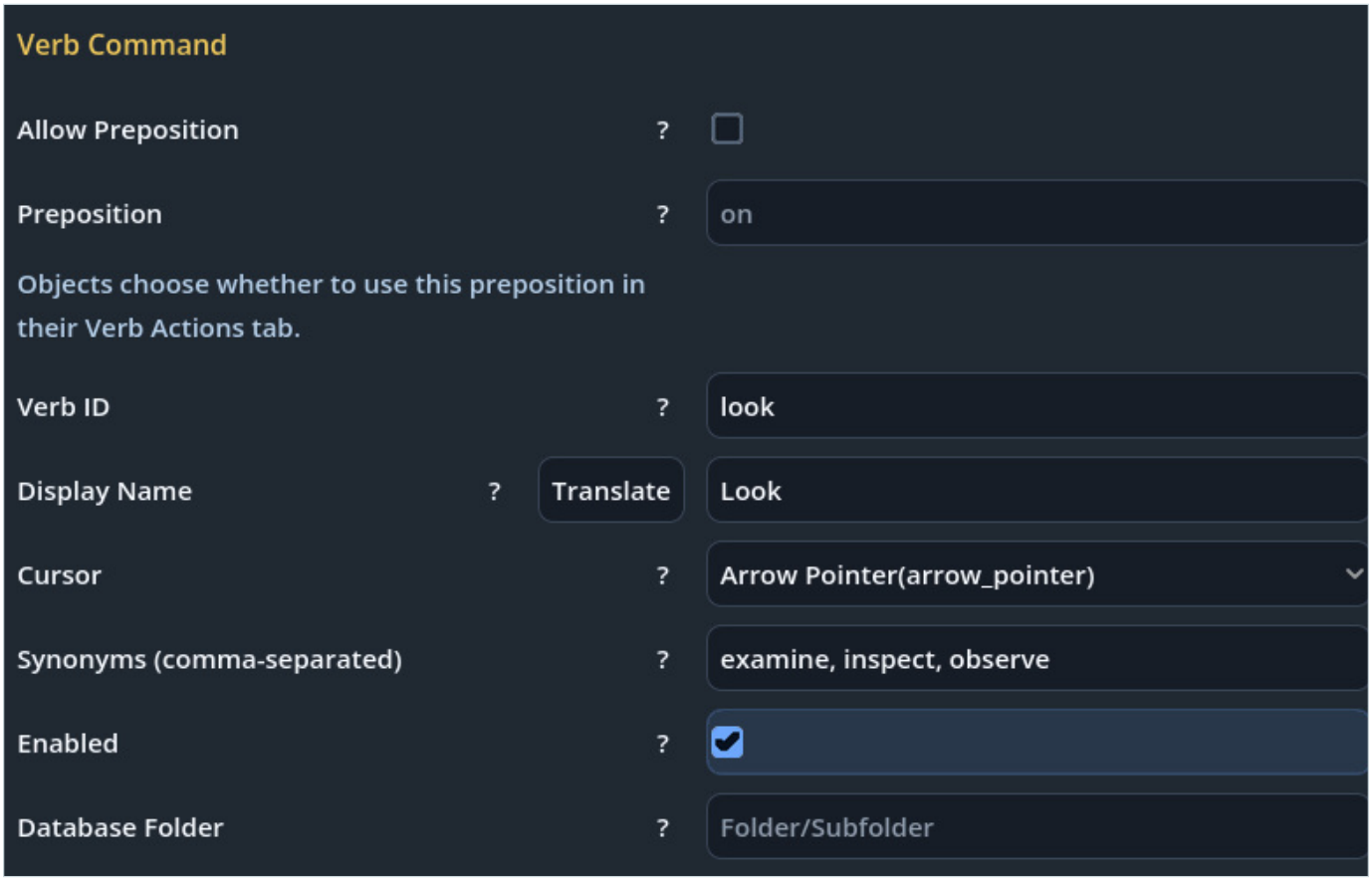


Figure 4. Look keeps a stable ID while examine, inspect and observe become alternative parser words.

A synonym is another way to request the same command. **Inspect** as a Look synonym does not provide a second, different response on the clock. Create a separate verb if Inspect genuinely needs different gameplay.

Use meaningful phrases and avoid giving the same short synonym to several verbs. A parser cannot use intention to resolve an ambiguous alias you authored.

Enabled and folders

Enabled controls the command's normal authored availability. Check custom GUI buttons as well; an existing button is a separate GUI element and is not automatically removed when you untick a verb.

Do not use "all verbs unchecked" as the way to suspend interaction. This build has legacy fallback verbs when there are no enabled authored definitions. The dedicated **Enable / Disable Verb Command Usage** action is the clearer runtime control for a cutscene or temporary lock.

Database Folder is project organisation. Putting Wind in a Clockwork folder does not restrict it to the workshop room.

Allow Preposition and Preposition

These support a command with a first and second object: **Use lantern on clock**, for example. The verb defines the connecting word; the first object opts in and supplies the allowed second-target response. Chapter 14 walks through it.

Leave the checkbox off for a simple command such as Wind Clock. Filling in a preposition word alone does not make every object a two-target interaction.

4. Create the Wind command

You can follow these steps with any small custom command. Choose a verb whose result is easy to see before you add animation, sound or a more involved puzzle.

1. Open the Verb Designer.
2. Click **Add New**.
3. Enter `wind` in **Verb ID**.
4. Enter **Wind** in **Display Name**.
5. Leave **Allow Preposition** off.
6. Enter **wind up, crank** in **Synonyms (comma-separated)**.
7. Keep **Enabled** checked.
8. If you already made a Hand Pointer cursor, select it. Otherwise leave Default Cursor temporarily and return after chapter 7.
9. Optionally enter **Clockwork** in Database Folder.
10. Click Apply Changes, then Save Project.

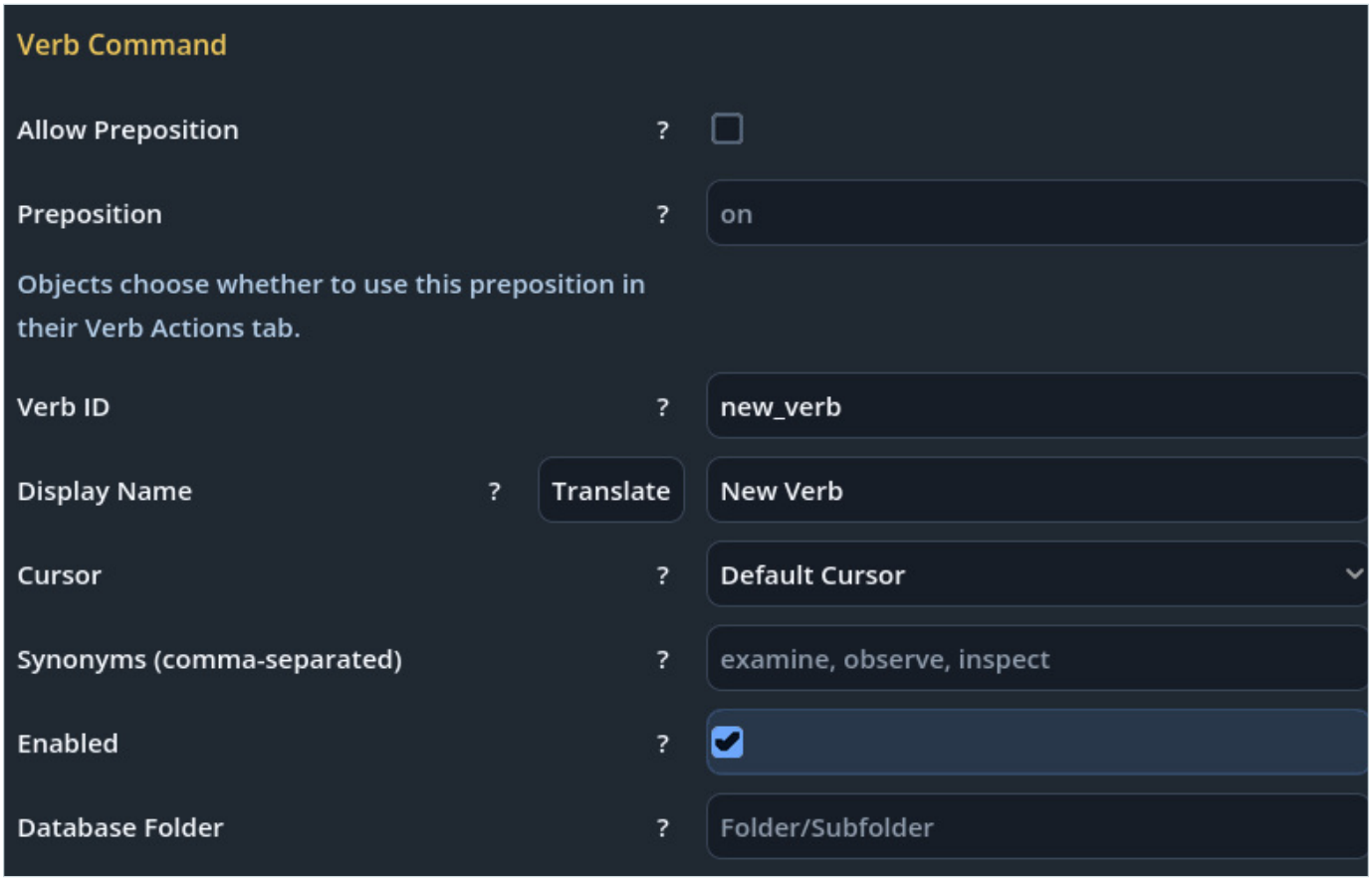


Figure 5. Add New begins with a new verb record. Replace the placeholder identity before using it in your adventure.

After applying, choose Wind in the top selector and check the stored values. There should be one Wind record with ID `wind`, not several copies whose names happen to look alike.

Creating the verb makes the command available to author. It does not yet define what winding a clock does. That belongs to the clock's response.

Add it to the offered verb set

1. Open **Global Settings**.
2. Go to **Adventure & Cursor**.
3. Find **Cursor Verbs**.
4. Open the selector and include `wind` along with the other commands you want available.
5. Apply Changes and Save Project.



Figure 6. Cursor Verbs is the configured set offered by the normal verb/cursor interface. Include the new wind ID.

When this list contains valid authored IDs, it filters the normal offered set. A valid old selection can therefore leave a newly created verb out of cycling or the built-in bar. A completely stale list has a fallback rather than deliberately making the interface empty.

This list is not a list of mouse image filenames. It contains verb IDs. Also, a custom GUI may have its own set of buttons; adding Wind here does not create a new button in your authored GUI.

5. Give the clock its response

In the Scene Designer, select the clock and use its inspector. You are now editing the target's behaviour, not the shared Wind definition.

1. Select your clock object in the scene or Scene Objects list.
2. Make sure it is enabled and its intended interaction area can be clicked.
3. Open **Verb Actions**.
4. Find **Wind Verb**. Scroll down if it follows several other command sections.
5. In its first Action selector, choose **Show Message** for the simplest test.
6. Enter **You wind the clock. A steady tick returns.**
7. Choose a duration such as **2.5 seconds**.
8. Apply Changes and Save Project.

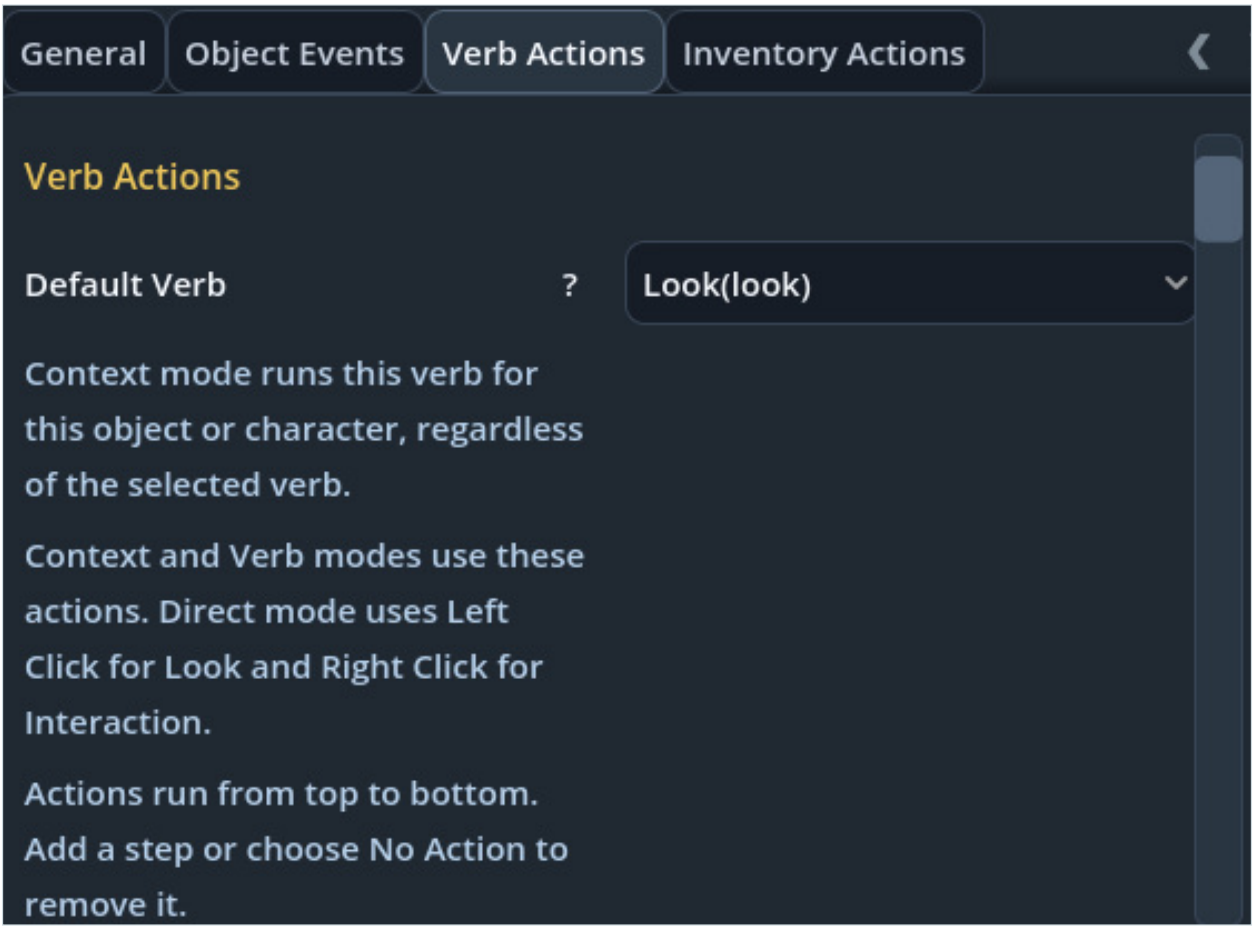


Figure 7. The target object has its own Verb Actions tab. Default Verb at the top is mainly relevant to Context mode.

Record a small piece of game state

The fuller example also records whether the clock has been wound. Create a Boolean game variable with ID `clock_wound`, initially false, in the Variable Designer.

Arrange the clock's Wind response in this order:

Step	Action	Values
1	Set Variable	Variable Clock Wound(clock_wound); Value `true`.
2	Show Message	The winding message; Duration 2.5 seconds.

Use **Add Next Action** in the Wind section to append another step. Make sure you are adding to Wind, rather than the similarly named button beneath another verb.

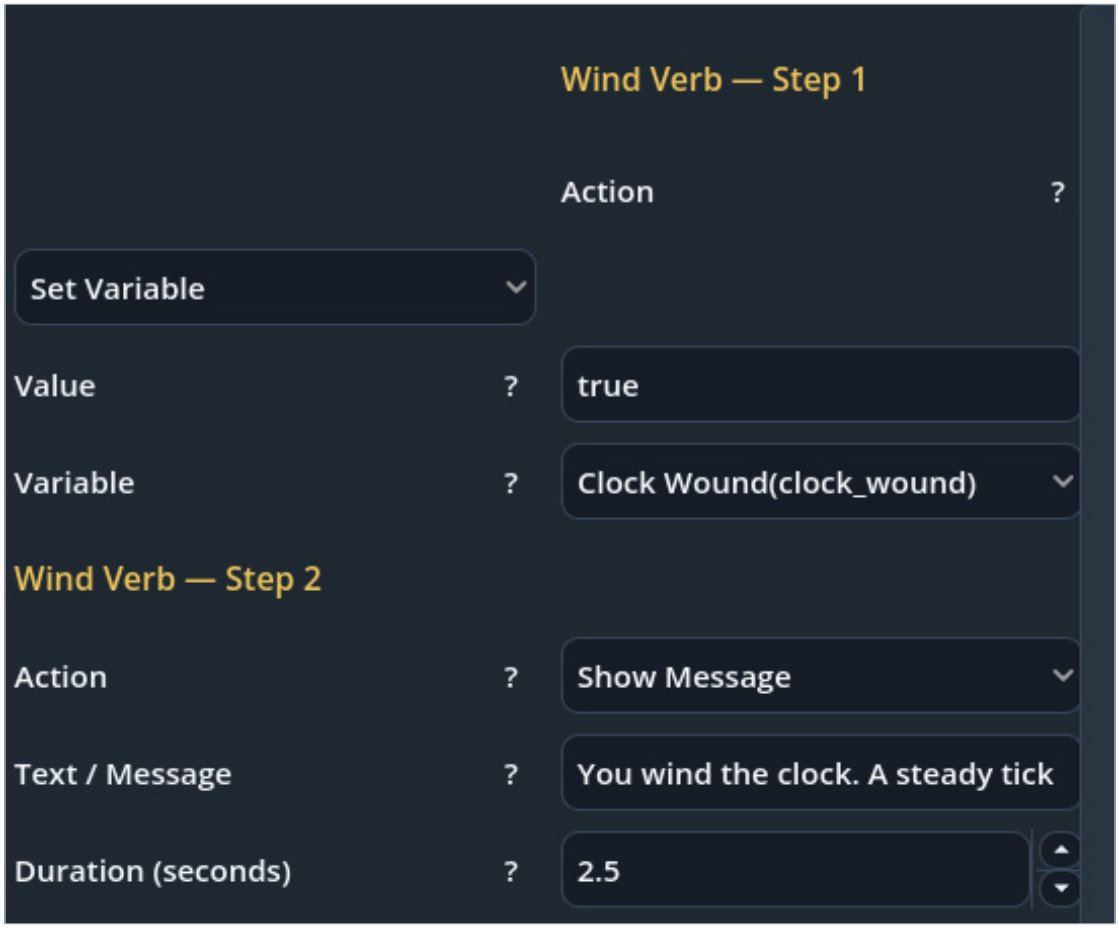


Figure 8. The clock's Wind response sets clock_wound to true, then shows its message. The actions run from top to bottom.

This demonstration records state and shows text. To make a clock model animate or play ticking audio, add the appropriate animation or sound action with a real animation/recording. A sentence mentioning a tick does not create the sound automatically.

Repeated winding will run the same simple response again. For a finished puzzle, use a condition in an action flow or script to distinguish the first use from later uses of `clock_wound`.

Default Verb is a separate decision

For this clock, set **Default Verb** to Look if a Context-mode click should describe it. Choose Wind if that same kind of click should wind it instead.

In **Verb Single Click**, the selected verb controls the interaction. Default Verb is not a replacement for the Wind response you just authored.

If a click takes another route

Check the object's **Object Events** tab. An authored click event can take priority over ordinary verb handling. If you deliberately use a click event but want it to run the current command, the action **Execute Selected Verb Command** provides that route.

Keep the first test simple: a clickable target, one Wind response, and no competing click event. Add extra input behaviour once that works.

6. Choose how the player selects commands

Open **Project Settings > General** and find **Interaction Mode**. This controls the order of the player's choices. The verbs and the object's responses remain useful across several modes.

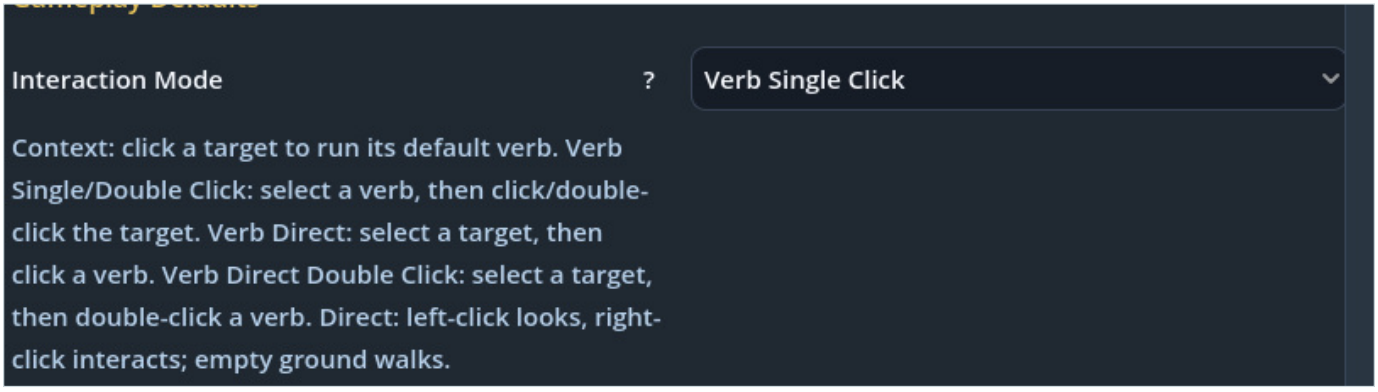


Figure 9. Interaction Mode is a project-level choice. Start with Verb Single Click for the step-by-step workshop example.

Mode	Basic sequence
Context	Click the target; its Default Verb chooses the response.
Verb Single Click	Select a verb, then click the target.

Mode	Basic sequence
Verb Double Click	Select a verb, then double-click the target.
Verb Direct	Select the target, then choose a verb.
Verb Direct Double Click	Select the target, then double-click the verb.
Direct	Left-click looks; right-click uses the object's interaction route; empty ground walks.
Cycling Cursors	Cycle the command/cursor, then use the selected command on a target.
Hotspot Then Verb Coin	Choose a target, then choose from the interaction coin.
Verb Coin Then Hotspot	Choose from the coin, then choose a target.

For the workshop steps, choose **Verb Single Click**, apply and save. Click Wind in the game GUI, then click the clock.

Right-click cycling

In Global Settings > Adventure & Cursor, **General Right Click Cycles Available Verb / Cursor Modes** allows an otherwise unhandled right-click to advance through the offered verbs in supported verb-oriented modes.

An object-specific right-click response can take priority. A selected inventory item also has its own right-click handling. Context, Direct and Verb Direct modes do not use this general cycle in the same way, so test the actual mode you chose.

Cycling and coin modes also depend on their corresponding Runtime Control Profile feature switches. If a requested mode behaves like Context, check whether the shared control profiles, cycling or coin features have been switched off.

Movement and interaction are different settings

Point-and-click movement, direct keyboard/gamepad movement and first-person movement describe how the character moves. Interaction Mode describes how the player chooses a command and target. A hand cursor alone does not change either system.

A gamepad profile may hide the mouse pointer and use proximity highlighting. Keep the commands understandable through labels and buttons as well as cursor art.

7. Understand every Cursor Designer field

Open the Cursor Designer and select an existing pointer to inspect it. The editor heading says **Cursor Designer**; these are the game cursor records used by verbs and pointer-feedback settings.

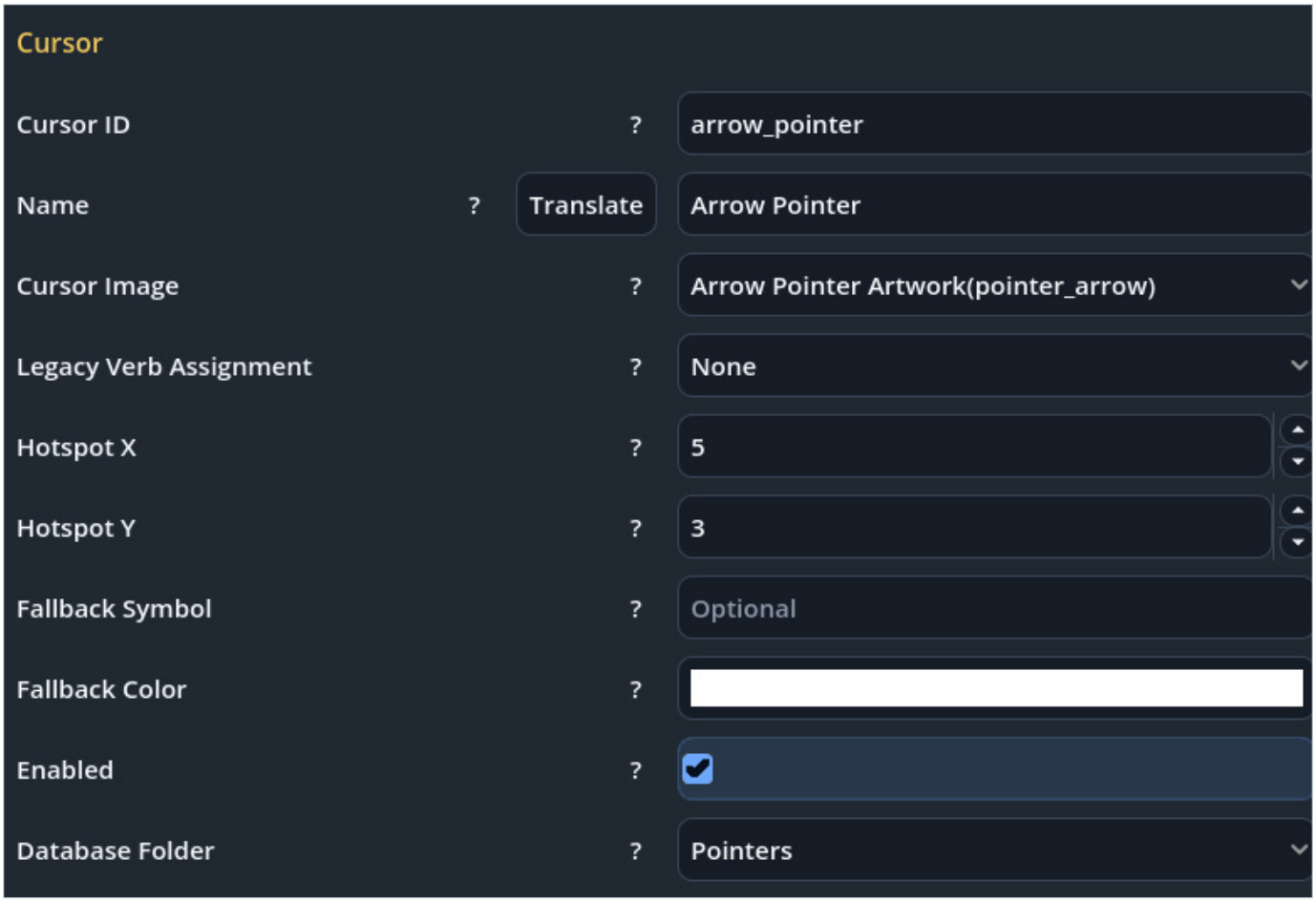


Figure 10. The Arrow Pointer record chooses an image and its hotspot. Its ID is separate from the image asset ID.

Field	Purpose
Cursor ID	Stable reference, such as `arrow_pointer`.
Name	Readable record name, such as Arrow Pointer.
Cursor Image	Imported image asset used for the pointer, or System Default Cursor.
Legacy Verb Assignment	Compatibility field for older cursor-led projects. Use the verb's Cursor selector for a new setup.
Hotspot X	Horizontal pixel coordinate of the click point inside the source image.
Hotspot Y	Vertical pixel coordinate of the click point inside the source image.

Field	Purpose
Fallback Symbol	A stored fallback field; see the current-build note below.
Fallback Color	Companion colour field for the fallback symbol.
Enabled	Allow this cursor record to supply a usable custom image.
Database Folder	Organise cursor records in the project.

Make a cursor record

1. Import a suitable pointer image through the Asset Importer.
2. Open Cursor Designer and click **Add New**.
3. Give it a unique Cursor ID and a readable Name.
4. Choose the imported image in Cursor Image.
5. Set Hotspot X and Y to the intended tip or centre of that image.
6. For a new project, leave Legacy Verb Assignment on **None** and link the cursor from the verb instead.
7. Keep Enabled checked.
8. Apply Changes and Save Project.
9. Return to the Verb Designer and assign this cursor to the relevant verb.

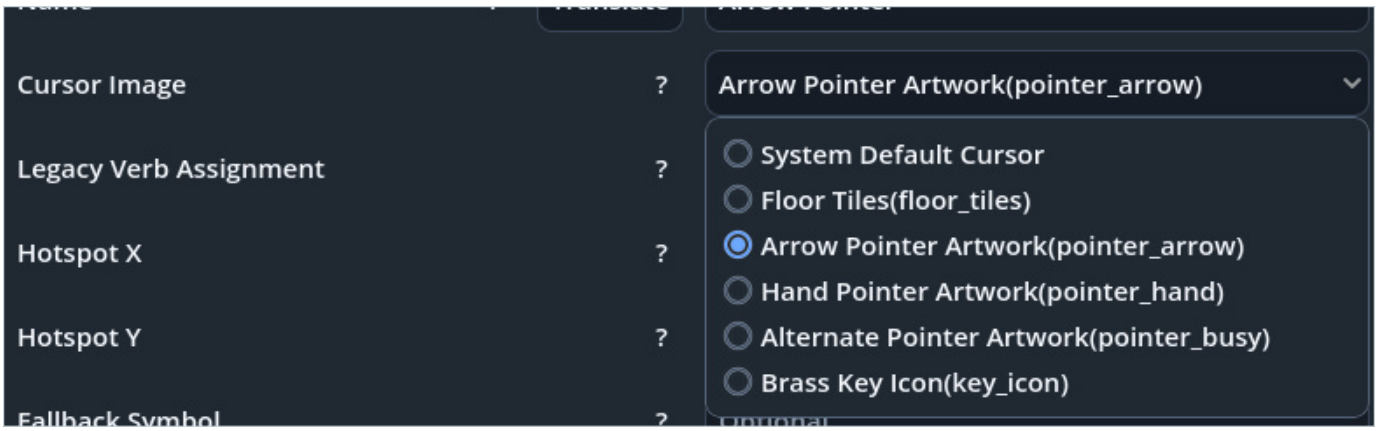


Figure 11. Cursor Image chooses an imported image asset. This is a different selector from the verb's Cursor field.

Several verbs can share one cursor. Several cursor records can also share one image while using different hotspots or names, though a clear naming scheme makes those cases easier to maintain.

About the fallback controls in this build

The form exposes Fallback Symbol and Fallback Color, but the current desktop cursor path uses a valid image or returns to the operating-system pointer. Do not depend on a typed fallback symbol becoming a working desktop pointer.

For a dependable release, supply a usable image or deliberately choose the system cursor, then test that choice. This designer does not present a cursor-animation frame list; do not assume an animated GUI image setup automatically becomes an animated mouse cursor.

8. Set the click point and pointer size

A cursor image has two things the player notices: the picture itself and the exact point that clicks. Those need to agree.

For an arrow, the click point is usually the sharp tip. For a crosshair, it is usually the centre. For a hand, it is usually the pointing fingertip.

Read hotspot coordinates

Coordinates start at the image's top-left corner. X increases to the right; Y increases downward. They refer to the source image's pixels.

Source image	Intended click point	Example hotspot
24 x 24 arrow artwork used here	Visible tip at pixel 5, 3	X 5; Y 3.
64 x 64 crosshair	Centre	X 32; Y 32.
Any hand image	Its actual fingertip	Measure that pixel in your image editor.

The crosshair row is a numerical example, not a cursor file you need to download. The workshop Arrow Pointer uses existing studio artwork with a tip at 5, 3.

Do not enter the object's screen coordinates or world position in these fields. The cursor hotspot does not resize a door's collider, the yellow selection bounds, the object's event volume or the walkable ground.

Test alignment on something small

1. Start the game with the custom cursor enabled.
2. Select the verb using that pointer.
3. Move its visible tip onto a narrow, easily recognised part of a target.
4. Check when the hover name or feedback appears.

5. Click and confirm the intended target responds.
6. If the pointer feels offset, correct the hotspot and retest.

Do not enlarge every object interaction area to compensate for a badly placed pointer hotspot. Check which of those two systems is actually misaligned.

Cursor Scale

Set **Cursor Scale** in Global Settings > Adventure & Cursor. It changes the rendered cursor image size; it does not change the cursor record's source-pixel coordinates.

The current desktop path fits the image's longest side to roughly 64 times this scale, within a 16-to-256-pixel range. Its proportions are preserved, and the hotspot is scaled with the image. Scale 0.6 therefore gives a longest side of about 38 pixels; scale 1 is about 64 pixels.

This is why a 24-pixel source image can appear larger in play. Keep the source artwork compact and clear, and check the actual result at your target display size.

The hotspot fields in this form allow values up to 256. Very large artwork with a click point hundreds of pixels into the file is awkward here. Prepare sensibly sized pointer artwork instead of using a full-screen image as a cursor.

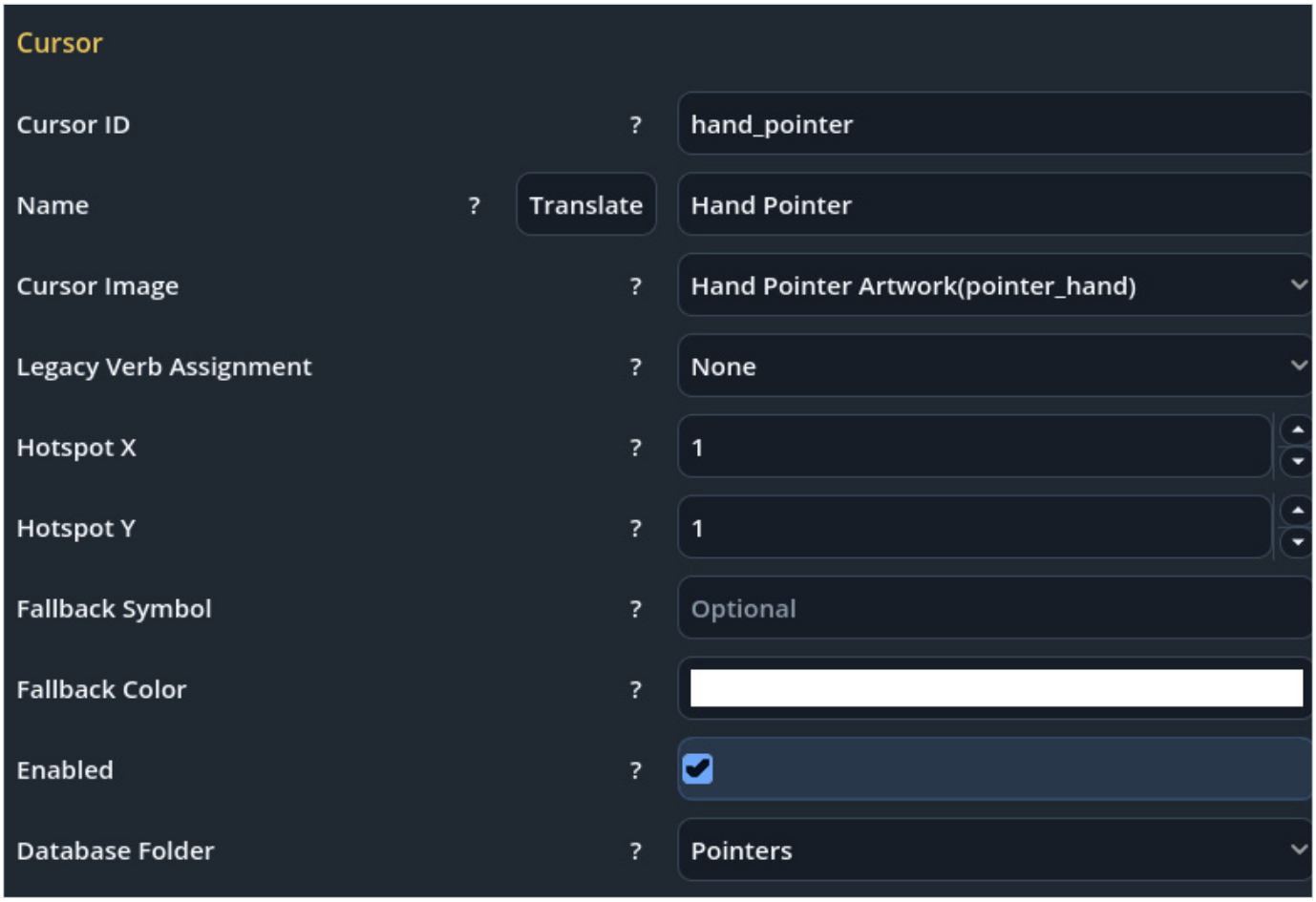


Figure 12. A second cursor can use different artwork and hotspot coordinates. Wind and Use share this Hand Pointer in the example.

9. Connect the global cursor settings

The Cursor Designer defines records. Global Settings decides how the game uses the cursor system as a whole.

Open **Global Settings > Adventure & Cursor** and scroll to Cursor Verbs and the cursor controls.

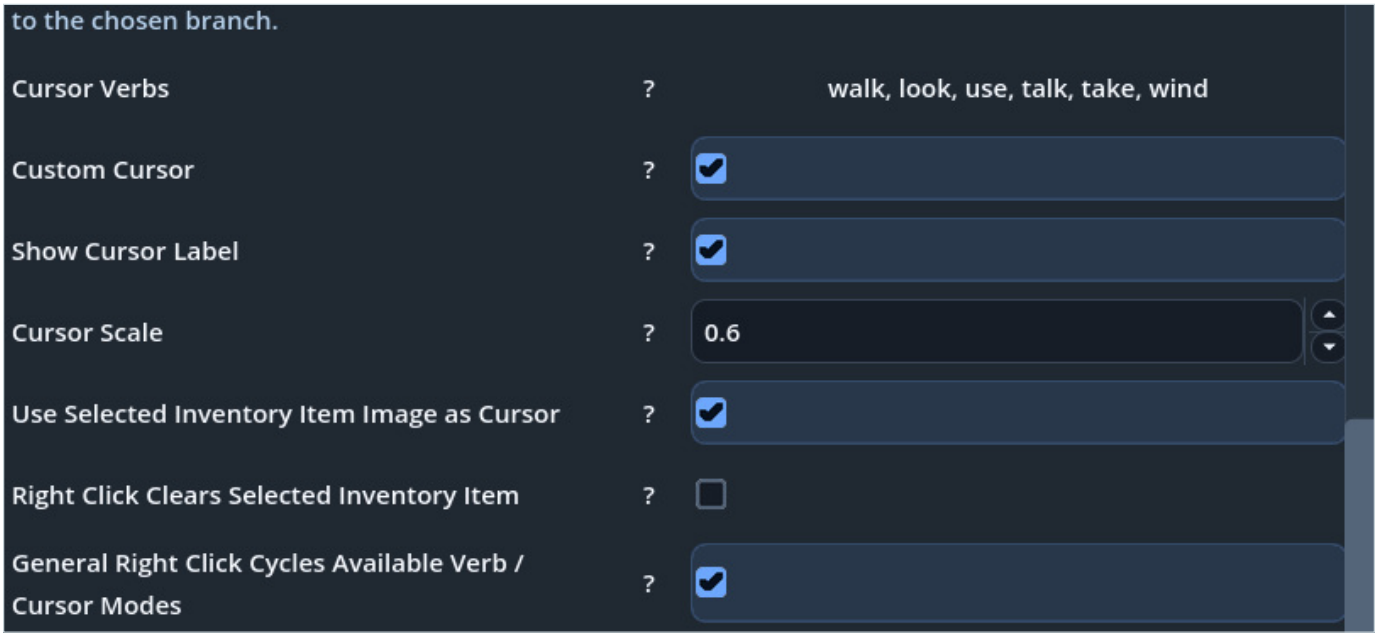


Figure 13. Global cursor controls include the offered verb set, custom pointer switch, size and selected-item behaviour.

Setting	What to check
Cursor Verbs	Include the authored verb IDs offered by the normal command interface.
Custom Cursor	Enable custom verb and object pointer-feedback images.
Show Cursor Label	A stored label preference; test the actual display in this build. A custom command label is a reliable explicit GUI choice.
Cursor Scale	Adjust the rendered image size as described in chapter 8.
Use Selected Inventory Item Image as Cursor	Display a selected item's icon as the pointer when that route applies.
Right Click Clears Selected Inventory Item	Give right-click the selected-item cancellation behaviour you want.
General Right Click Cycles Available Verb / Cursor Modes	Allow cycling on otherwise unhandled right-clicks in the supported modes.

Apply Changes and Save Project after editing these settings.

Which cursor takes priority?

While the mouse is inside the active game viewport, the current desktop path tries these routes in order:

1. A valid object press/hover cursor, when Custom Cursor is enabled. A press cursor takes precedence over hover while held.

2. A selected inventory item's icon, if its separate item-cursor setting is enabled.
3. The current verb's cursor, when Custom Cursor is enabled. A Change Game Cursor override affects this route.
4. The operating-system pointer if no usable custom image is supplied.

When the pointer is over ordinary GUI controls or outside the active game preview, the normal interface pointer can be restored. Move into the room when checking the selected verb's cursor.

A forced cursor does not necessarily beat an object's hover override or a selected inventory icon. Check the active route before changing the wrong record.

The inventory-item option is separate from Custom Cursor. Turning Custom Cursor off does not, by itself, promise to turn the selected-item icon route off too.

Inventory icons as pointers

When an item is selected, its icon can be used as the pointer. That is useful for a key-on-door style interface. This route centres the hotspot on the item image rather than using an unrelated cursor record's hotspot.

It still needs the actual inventory interaction response on the target or item. Displaying a key beside the mouse does not automatically unlock a door.

Check the game, not only the form

The form contains some compatibility-oriented controls. In particular, do not rely on Show Cursor Label creating a floating pointer caption in every display path. If your interface needs to tell the player the current command, add a GUI label with **{command}**, as shown in chapter 11.

10. Give an object hover and pressed cursors

An object can use pointer-feedback cursors independently of the selected command. This is useful when you want a target to signal that it is interactive, or to show a brief change during a press.

1. Create the cursor records you want to use.
2. Select the object in the Scene Designer.
3. Open **Pointer Feedback** in its inspector.
4. Choose **Hover Cursor**, **Left Click Cursor** or **Right Click Cursor**, as appropriate.
5. Apply Changes and Save Project.
6. Test entering, pressing, releasing and leaving the target.

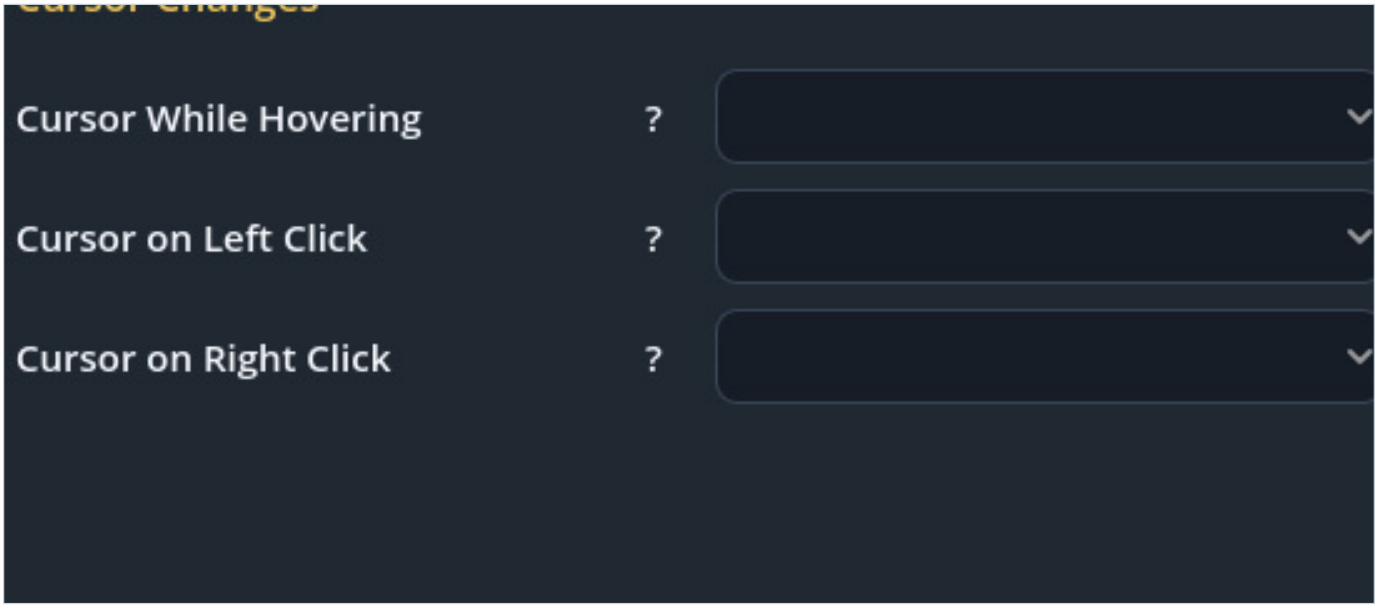


Figure 14. Object Pointer Feedback offers separate hover and mouse-button cursor references. These select cursor records, not button background pictures.

The pressed cursor is temporary. On the matching button release, the press state clears and the normal priority rules choose the next pointer. That may be the object's hover cursor while the mouse is still over it, rather than the verb cursor you see on empty ground.

A hover or pressed cursor only changes feedback. To make a mouse-button event run gameplay, author the appropriate action in Object Events or the normal Verb Actions route.

Three images that are easy to confuse

Image choice	Where it appears
Cursor record's Cursor Image	The mouse pointer.
Object's Hover / Left Click / Right Click Cursor	The mouse pointer while interacting with that object.
GUI element's background / hover / pressed image	The GUI element itself, such as a button face.

Changing the Wind button's hover image does not assign Wind's mouse cursor. Assign the cursor in the Verb Designer, and author the button's appearance in the GUI Designer.

Test the release as well as the press

Press over the object, move away while holding the button, then release. The pressed appearance should not remain stuck. Also test both mouse buttons if both have overrides.

If the pointer looks different over the clock and over empty ground, that may be the hover override working as configured. If you want only the selected verb's image, leave the object-specific cursor fields empty.

11. Add verb buttons and a command label to a GUI

A verb-selection button should choose the command the player intends to use. The target object still owns the response.

Make the Wind button

1. Open your game GUI in the GUI Designer.
2. Add a Button element and name it clearly, for example **Choose Wind** with ID `choose_wind`.
3. Set its visible text to **Wind**.
4. Open its **Events** tab.
5. Under **Click**, choose behaviour **set-verb**.
6. Choose **Wind(wind)** in the verb field that appears.
7. Save the GUI changes and Save Project.
8. Use Play / Restart and click the real button in the running game.



Figure 15. The Wind button uses the dedicated set-verb behaviour and selects Wind(wind). It does not contain the clock's winding response.

Use this dedicated behaviour for a normal verb-selector button. It follows the game's verb-selection route, including the pending-target sequence used by Verb Direct modes. The general **Change Verb** action changes the current selection; it does not by itself reproduce every part of that GUI selection sequence.

Do not put the winding message directly on the Wind button if the player still needs to choose which object to wind. Otherwise the button would run the response before a target is selected.

Show what the player has selected

1. Add a Label element to the same GUI.
2. Give it a useful ID, such as `command_label`.
3. In its normal text field, enter **{command}** exactly, including braces.
4. Place it where the player can read it without covering the scene.
5. Run the game, select a verb and move over a target.

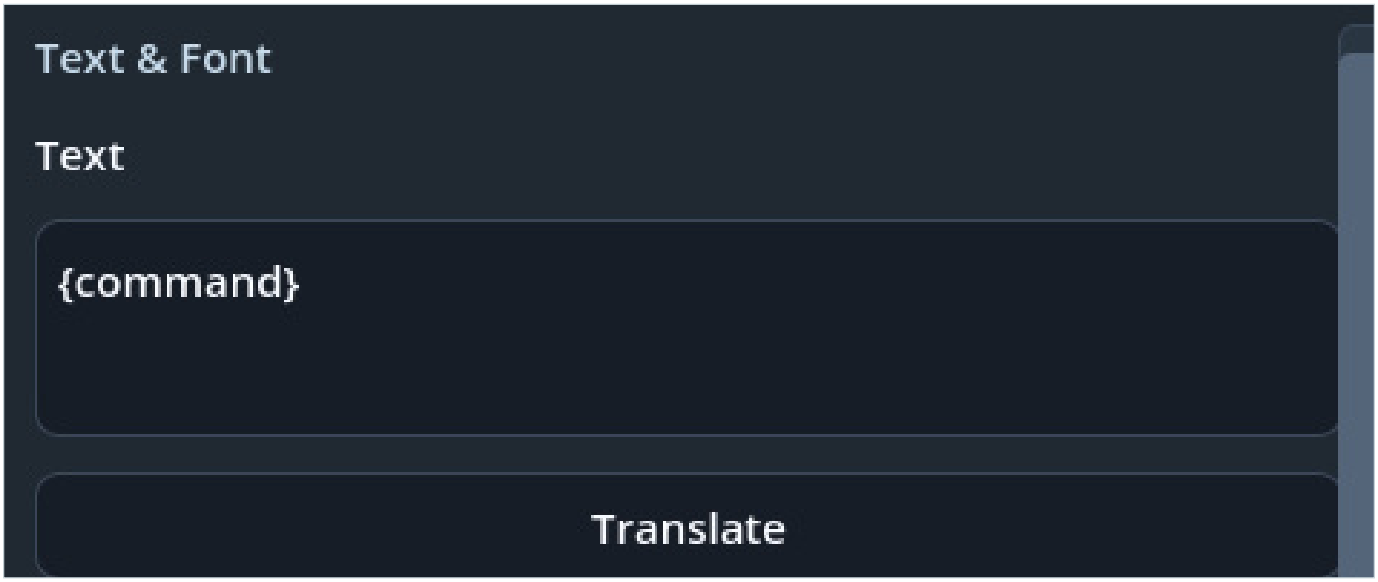


Figure 16. The label's authored text is {command}. The game fills it with the current verb and hovered target.

GUI text marker	Example meaning
`{verb}`	The current command name or active two-target prefix.
`{target}`	The current hovered target label.
`{command}`	The command/prefix followed by the hovered target.

For example, the label can show **Wind Workshop Clock**. During an opted-in preposition command, the prefix can become **Use Bench Lantern on** while the game waits for the second object.

These are the GUI text markers for this job. Do not substitute an invented `@Verb` token. Other game-text placeholders have their own syntax and purpose.



Figure 17. The running GUI has selected Wind. Selecting the command does not yet execute the clock's response.

Assign the intended GUI

If your GUI is meant to replace the main game HUD, assign it in **Global Settings > System GUIs > Main HUD**. An assigned custom HUD can hide the built-in verb bar, so include the controls and command label your players need.

A visible GUI can also be part of another screen arrangement. Check its activation and visibility rules. Creating a button inside an invisible or unassigned screen does not make it appear in the game.

The designer's preview is useful for layout and image states. Check gameplay events in the running preview.

12. Give the text parser the same vocabulary

The parser gives the player another route into the same commands. A mouse player can choose Look and click a clock. A typing player can enter **please examine the timepiece**.

Three authored choices make that sentence work:

Sentence part	Where it is defined
examine	A synonym of the Look verb.
timepiece	A synonym of the clock object.
please, the	Words removed by the parser's ignored-word settings.

Give the target a useful synonym

Select the clock in the Scene Designer and find its **Synonyms** field in the object's general settings. Add **clock, timepiece, grandfather clock** if those words suit your object. Apply Changes and Save Project.

Verb synonyms identify commands. Object synonyms identify targets. Putting **timepiece** in Look's synonym field would mix up those jobs.

Use Command Preview

1. Open **Text Parser Designer** from its toolbar icon or project workspace.
2. Check **Enable Text Parser**.
3. Review **Words to Ignore**. Words such as a, an, the, please and at are useful starting points.
4. Click **Apply Parser Settings** after changing the parser settings, then Save Project.
5. In **Command Preview**, enter **please examine the timepiece**.
6. Read the cleaned command, matched verb and matched target.

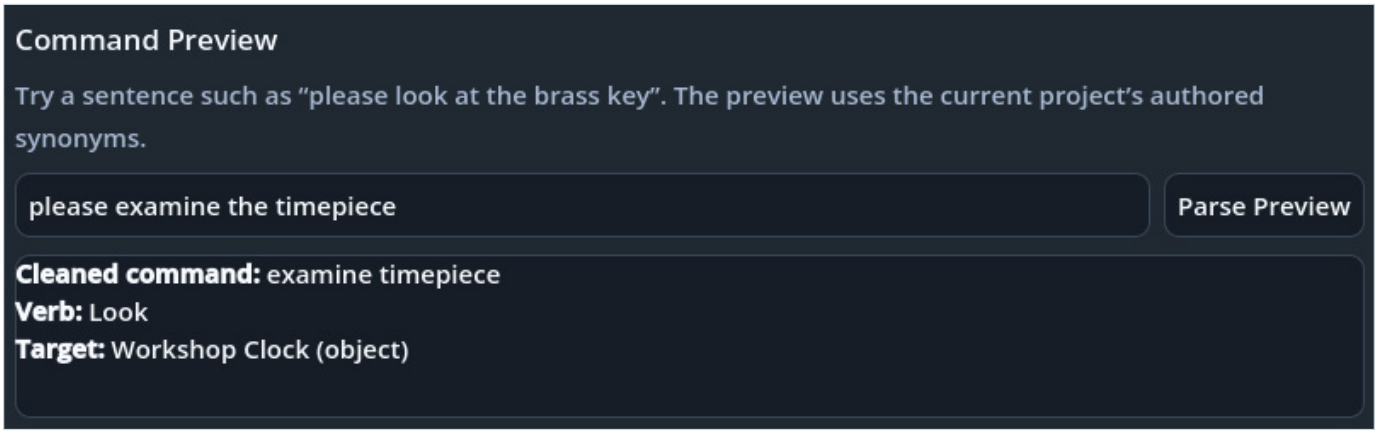


Figure 18. Command Preview removes ignored words and recognises Examine as Look and Timepiece as the workshop clock.

Previewing a command shows how it is recognised. It does not execute the clock's actions, move the player, or prove that a gameplay interaction is currently reachable.

Now enter **wind up the clock**. The phrase **wind up** is a Wind synonym, so it should identify the same custom verb as the GUI button.

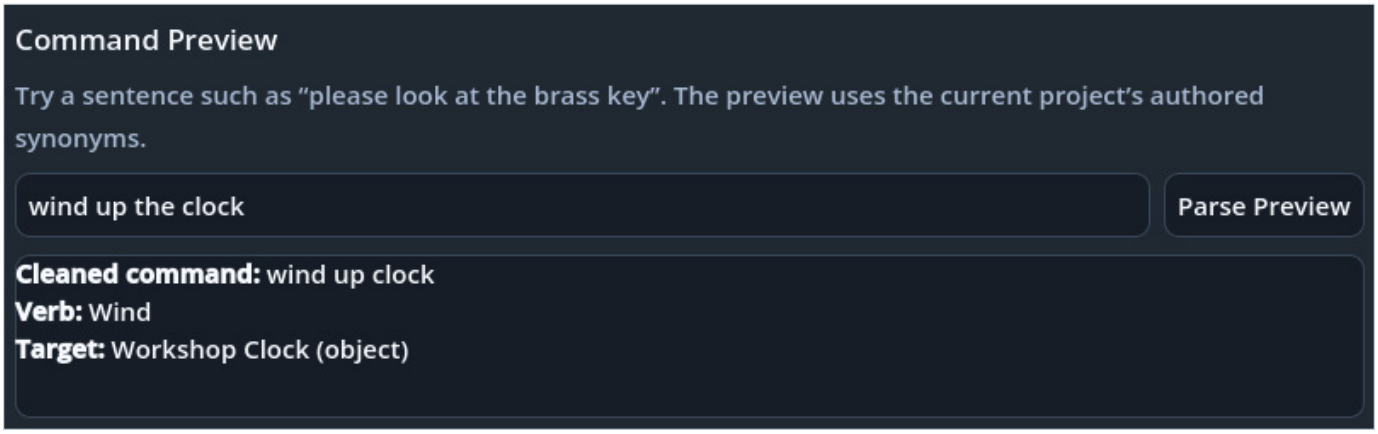


Figure 19. A multiword synonym can request Wind. The target still needs its authored Wind response.

Keep the vocabulary predictable

Recognition ignores normal capitalisation and cleans common punctuation. Prefer distinctive target names and avoid making several nearby objects share a broad alias such as **thing**.

Longer matching aliases are preferred. When equally good aliases collide, the first matching record can win. Do not use that as puzzle logic; choose clearer names and synonyms.

The runtime resolves targets from the active room and the relevant item/region records. The editor preview can see authored project information without representing the exact active-room state. Test a target from the room where the player will use the command.

Custom Commands are a separate route

The parser can also match a complete custom phrase and run its assigned script. For example, **help** can call a short help script. This route is checked before the normal verb-and-target route.

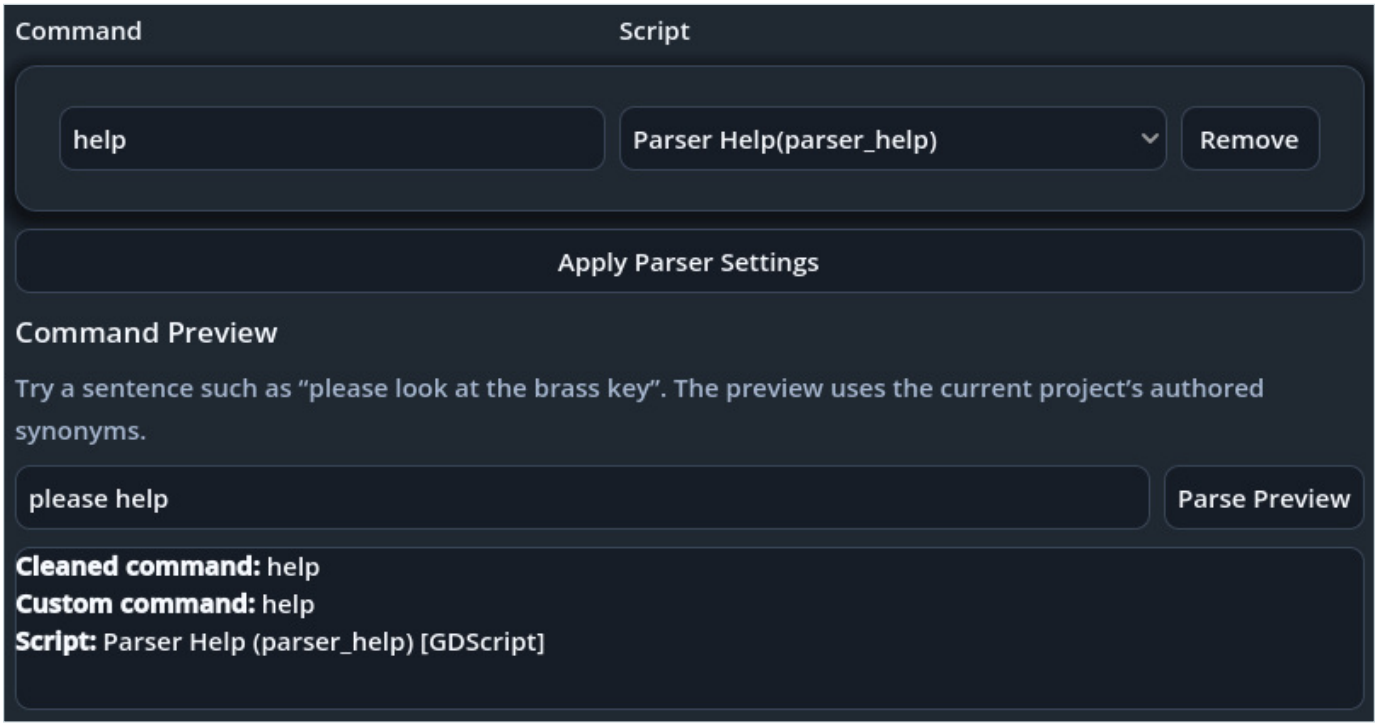


Figure 20. A complete custom Help command points to a script. It is a different arrangement from defining Help as an ordinary target verb.

Use custom commands for deliberate whole commands. Do not create one custom entry for every "Look at object" sentence when normal verbs and object synonyms already provide that behaviour.

Translated display text does not automatically create a complete vocabulary for a parser in another language. Supply the aliases and command phrases that language needs, then test them in context.

13. Put a parser input box in the GUI

The parser settings define recognition. The player also needs somewhere to type.

1. Open the intended game GUI in the GUI Designer.
2. Add a **Text Box** element.
3. Give it an ID such as `parser_input`.
4. Keep **Multiline** off for this command-entry box.
5. In **Type Settings**, enable **Run Text Parser Command When Enter Is Pressed**.
6. Optionally enable **Clear Single-Line Text When Enter Is Pressed**.
7. Enable **Return Focus To Text Box After Enter** if repeated keyboard entry suits the game.
8. Give it a useful placeholder, such as **Try: please examine the timepiece**.
9. Save and test it in the running game.

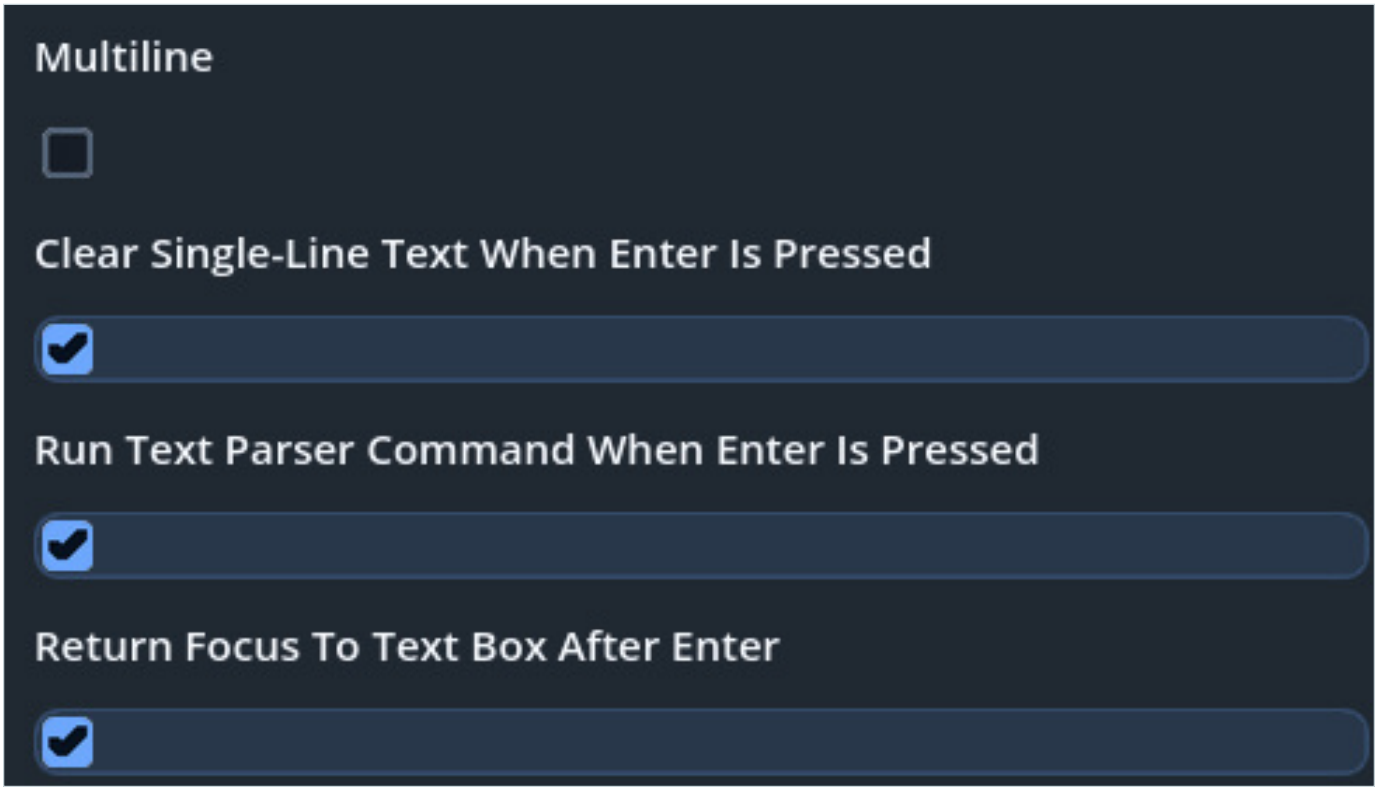


Figure 21. The single-line text box can submit its contents to the parser on Enter. Clear and focus-return behaviour are separate choices.

The placeholder is a prompt, not a submitted command. The player types the actual sentence, then presses Enter.

Keeping focus helps a typing-led game. In a mixed mouse/keyboard adventure, also check that the player can click back into the room and that movement keys are not unexpectedly being entered as text.

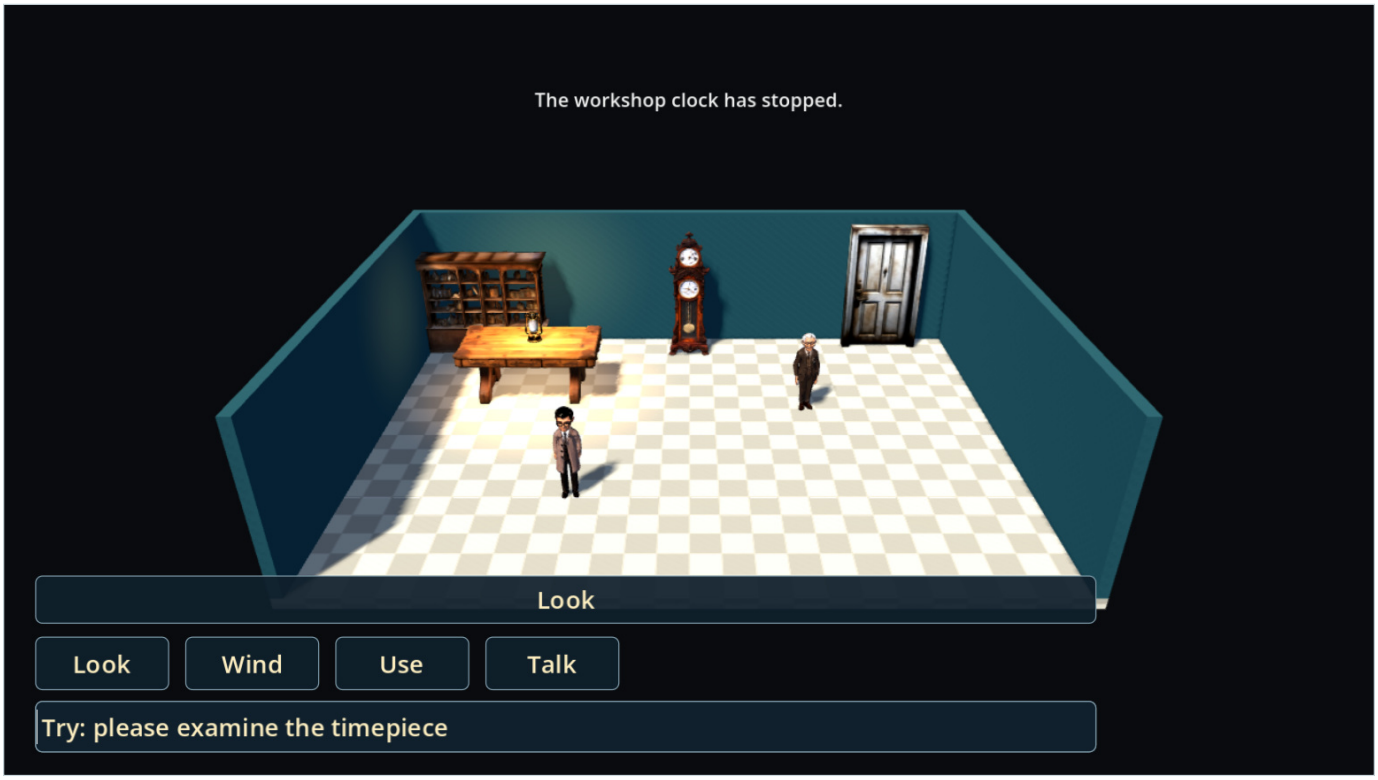


Figure 22. Submitting "please examine the timepiece" through the GUI text box reaches the clock's Look response.

You do not need to run a second script just to duplicate the checkbox's parser submission. If Enter produces the same response twice, check whether you have also added another submit event that calls the parser.

14. Use one object on another

A preposition command has a source, a connecting word and a second target. Here the player uses the bench lantern on the clock to see a winding socket.

Set up the verb

1. Open the **Use** verb in the Verb Designer.
2. Keep the ID use.
3. Check **Allow Preposition**.
4. Enter **on** in **Preposition**.
5. Apply Changes and Save Project.

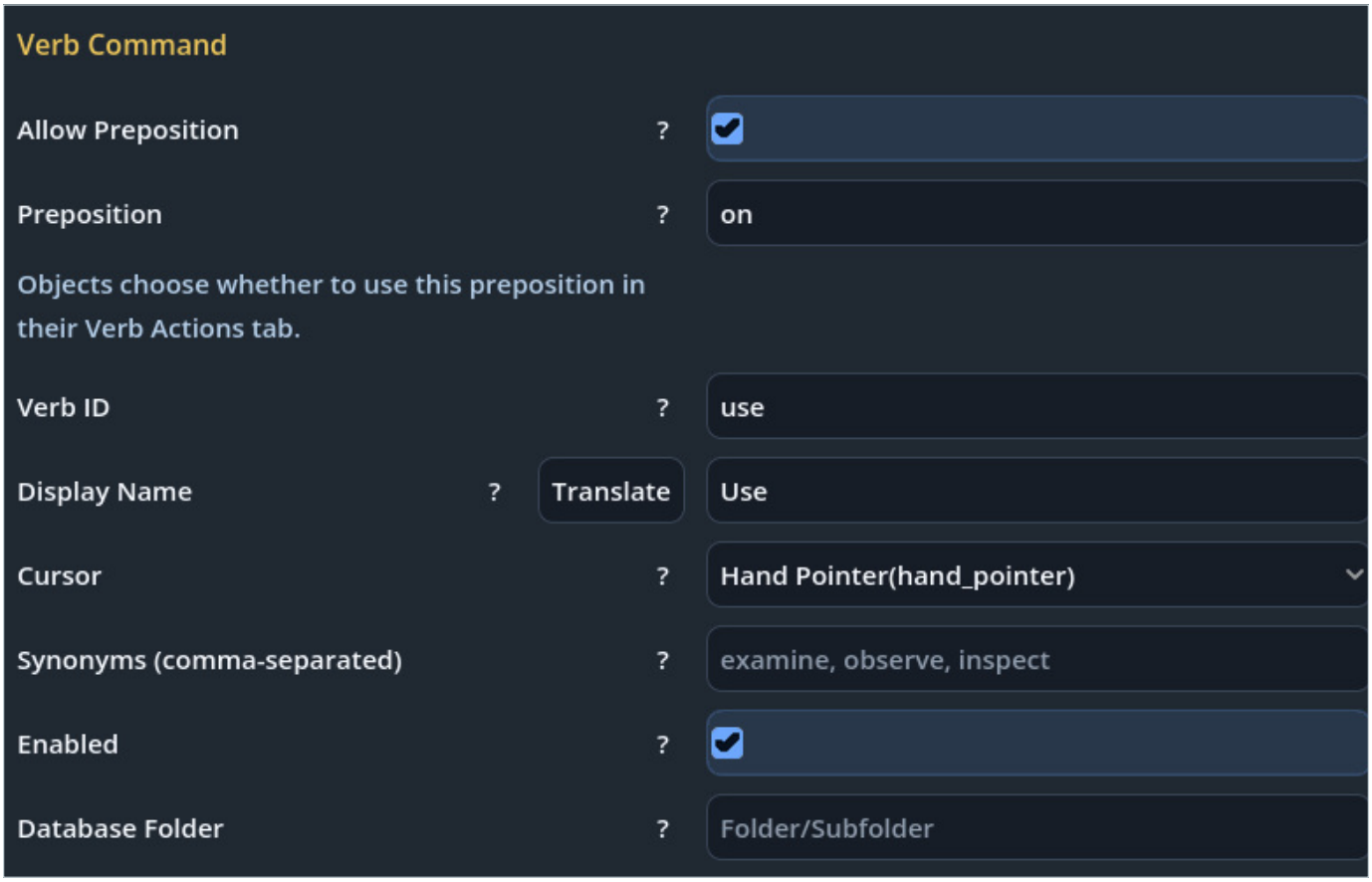


Figure 23. Use defines "on" as its connecting word. Individual source objects still choose whether to use this two-target form.

Opt the first object in

1. Select **Bench Lantern** in the Scene Designer.
2. Open its **Verb Actions** tab.
3. Check **Use Preposition - Use on**. The label includes the selected verb and preposition wording.
4. Find **Second-Target Responses** and click **Add Second-Target Response** if there is not already a row.
5. In that response, choose Use as the Verb.
6. In **Allowed Targets**, select **Workshop Clock**.
7. Set its Result Action to Show Message and enter **The lantern reveals a tiny winding socket.**
8. Apply Changes and Save Project.

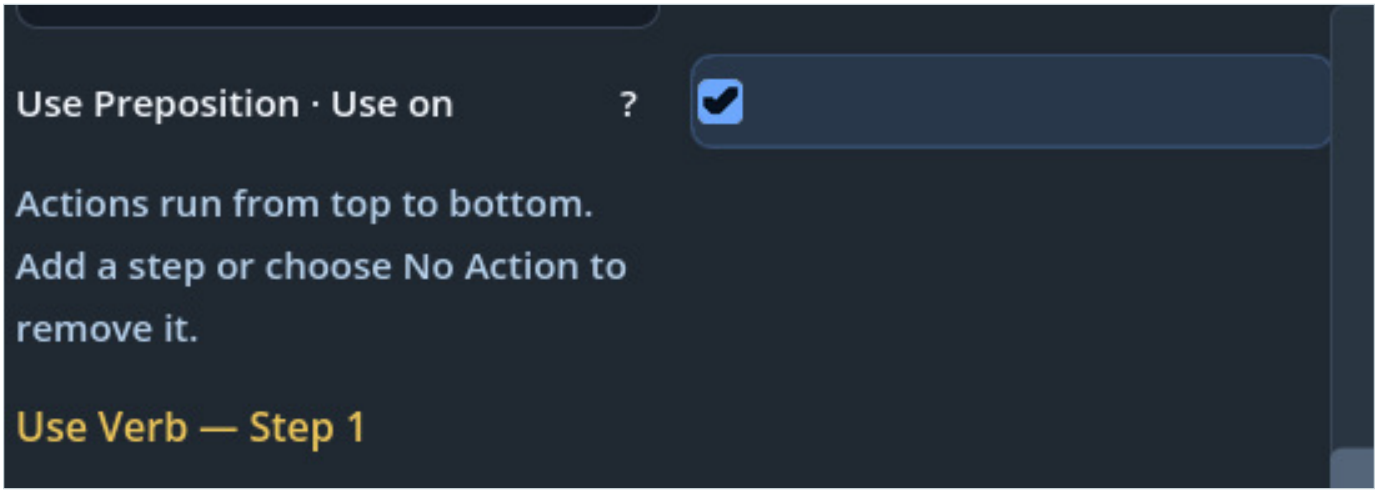


Figure 24. The source object opts in to the Use preposition. Other objects can continue to use ordinary one-target responses.

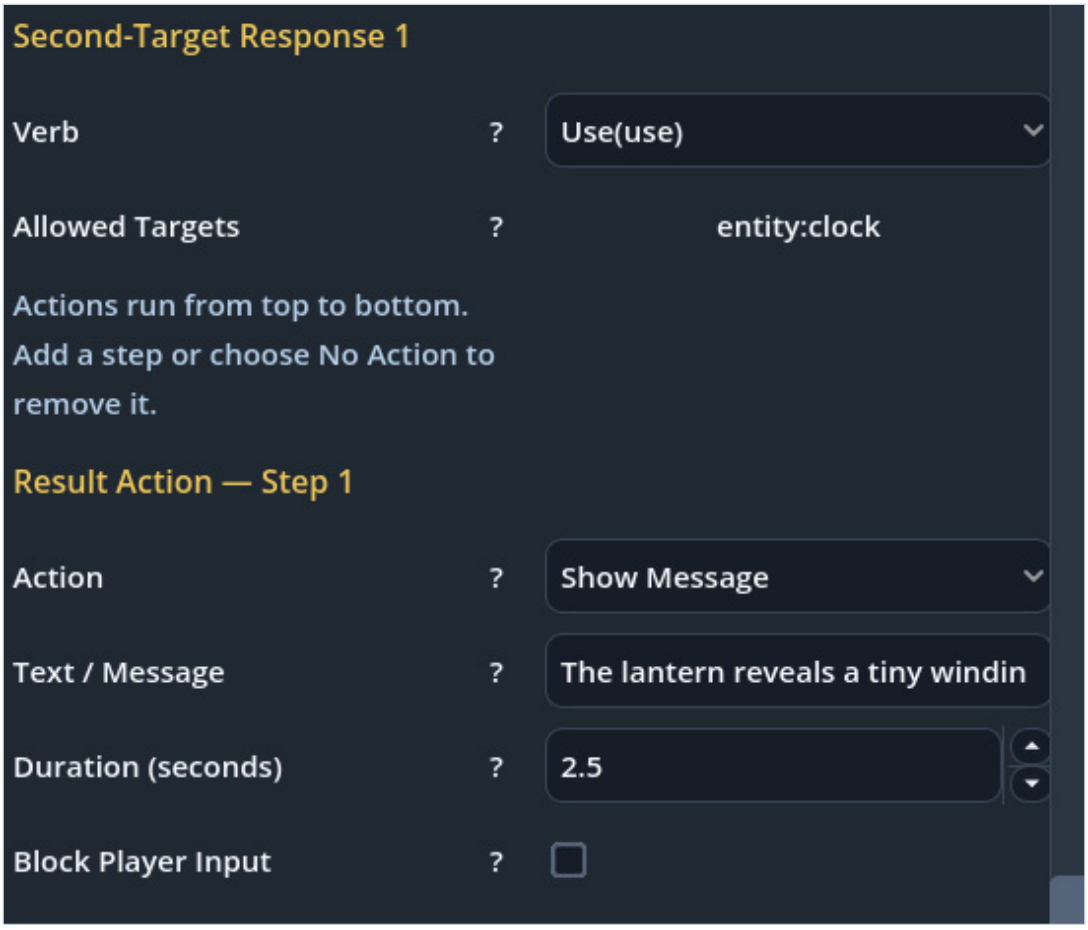


Figure 25. The response belongs to the first object, the lantern, and names the clock as an allowed second target.

Try it with the mouse

1. Start the game in Verb Single Click.
2. Choose Use.
3. Click Bench Lantern.
4. Check that your command label now reads **Use Bench Lantern on**.
5. Click Workshop Clock.
6. Check the two-object response.

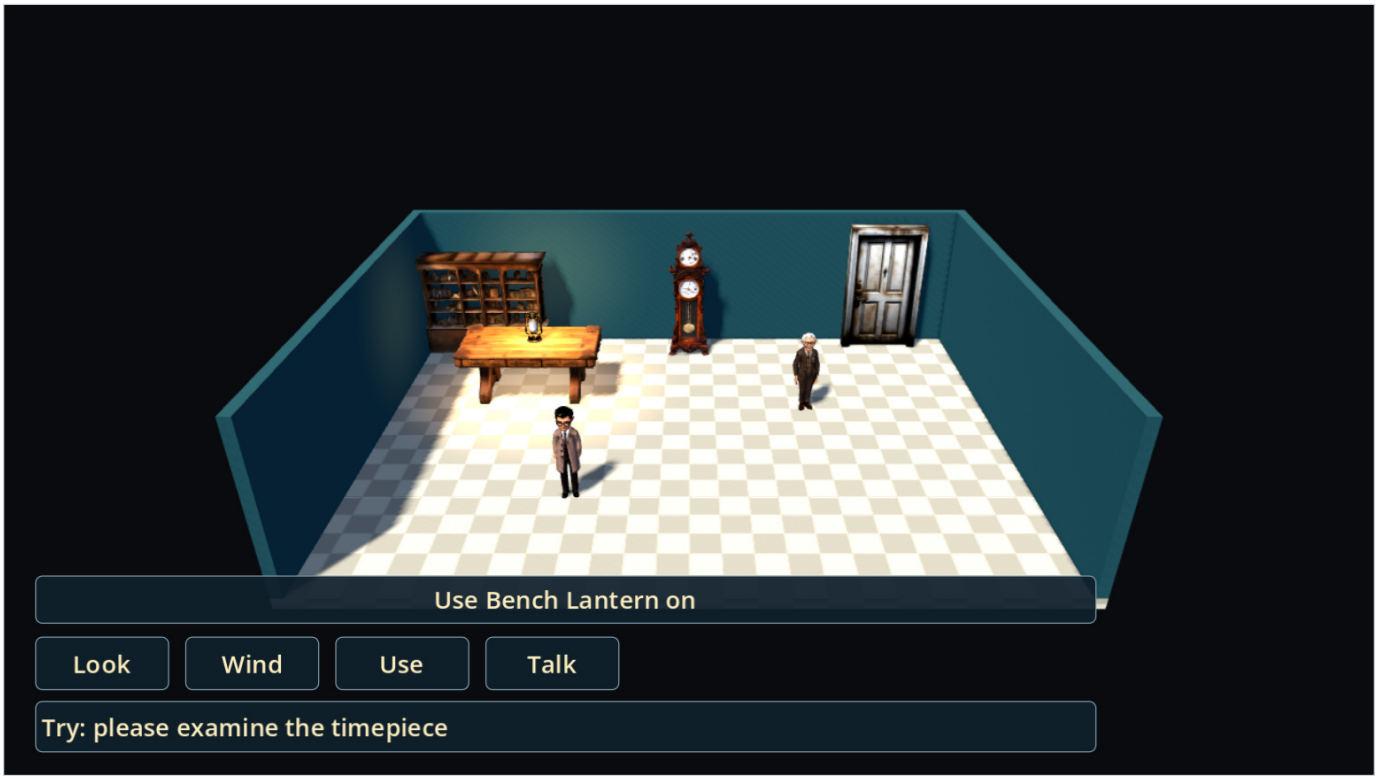


Figure 26. The first object is selected and the game is waiting for the second one. The command label makes that state visible.

The source must be enabled, visible and valid. For an inventory item, the player must actually carry an eligible item. A selected name alone does not bypass those checks.

Try the typed version

Enter **use the lantern on the clock**. The runtime reads the configured connecting word before removing ordinary ignored words, so "on" can still separate the objects even if it is also on the general ignore list.



Figure 27. The typed two-object command reaches the lantern-on-clock response.

Keep a command like this straightforward. The current pair parser expects a complete object match on each side of the separator. Extra words or vague target names can prevent a valid pair.

The basic editor Command Preview is useful for ordinary alias matching; it does not display the whole two-target execution flow. Verify the pair in the running game.

Changing the selected verb cancels the pending pair. The shared **CancelPreposition** action is also available when you need an explicit cancel control. If a second target is invalid, the first selection may remain active for another try.

15. Change verbs and cursors through actions or scripts

Use the shared action system when a cutscene, GUI action list or visual script needs to change a current command or pointer.

Action	Main fields	Result
Change Verb	New Verb (`verbId`)	Select a command.
Change Game Cursor	Cursor (`cursorId`)	Request a cursor override without changing the selected command.
Enable / Disable Verb Command Usage	Enabled (`enabled`)	Suspend or restore the shared verb interaction route.
Execute Selected Verb Command	Character / Object (`entityId`)	Run the currently selected command on a target.
CancelPreposition	None	Cancel a pending first-object selection.

A cursor override changes the pointer route, not the meaning of a click. You can show an alternate cursor while the selected command remains Look.

GDScript example

Put these calls inside the entry function supplied by the script editor. Keep the template's existing wrapper and parameters.

```
await ChangeVerb({"verbId": "wind"})
await ChangeGameCursor({"cursorId": "alternate_pointer"})
```

To release a forced cursor and return to the normal selection rules:

```
await ChangeGameCursor({"cursorId": ""})
```

Do not change the verb merely to restore an overridden pointer. They are separate settings.

PCA Script

```
function run()
    ChangeVerb({"verbId": "wind"})
    ChangeGameCursor({"cursorId": ""})
end
```

BASIC

```
FUNCTION Run()
    AWAIT ChangeVerb({"verbId": "wind"})
    AWAIT ChangeGameCursor({"cursorId": ""})
END FUNCTION
```

Lua

```
local Script = {}
function Script.run()
    await(ChangeVerb({verbId = "wind"}))
    await(ChangeGameCursor({cursorId = ""}))
end
return Script
```

Temporarily suspend verb use

Use **Enable / Disable Verb Command Usage** with false at the appropriate point, and restore it with true when interaction should resume. This guards the shared verb route. Keep the rest of your input/cutscene design consistent; it is not a promise to disable every unrelated custom script or GUI action.

When using a normal GUI verb selector, prefer **set-verb**, as explained in chapter 11. When you need a scripted selection or an explicit sequence of actions, use Change Verb and, where appropriate, an Execute Selected Verb Command action with a known target.

16. Check the whole chain in play

Use a fresh preview after applying and saving the authored records. Start with Verb Single Click and a visible command label; that makes each stage easy to observe.

Mouse route

1. Click Wind in your GUI.
2. Check that the command label changes to Wind.
3. Move from the button into the room and check the custom pointer.
4. Hover over the clock and check the target name.
5. Click the clock and confirm its Wind message.
6. Check `clock_wound` in your own debugging or game-state display if you added the variable step.



Figure 28. A close-up of the actual Hand Pointer inside the room. Moving over GUI controls can restore the normal interface pointer.

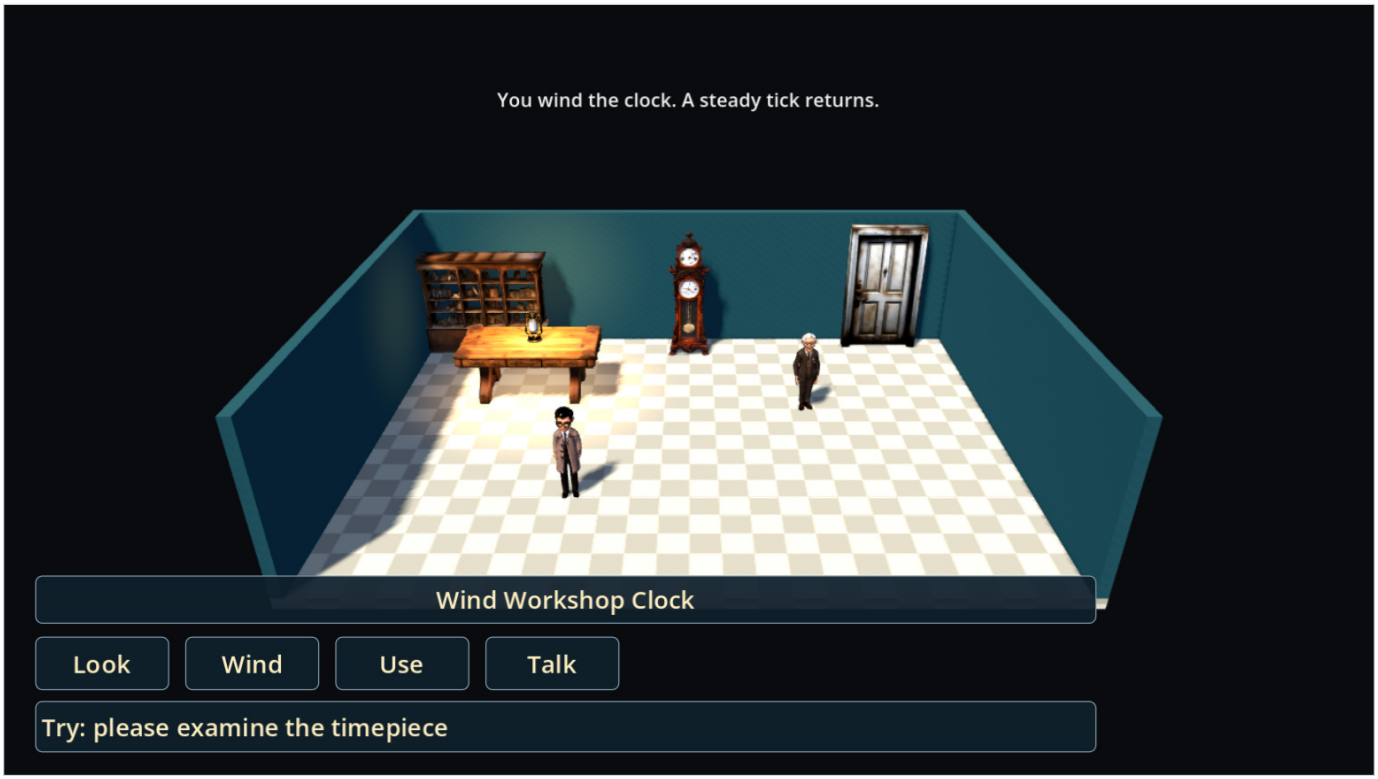


Figure 29. A mouse click on the clock after choosing Wind reaches its authored response.

Keyboard route

Enter **please examine the timepiece**, then **wind up the clock**. The first should select Look and describe the clock. The second should select Wind and run the same Wind response used by the mouse route.

Try one unknown verb and one unknown target as well. The response should make the problem clear enough that the player can try again.

Pointer feedback route

Test a target with a hover cursor. Press and release each configured mouse button, then move away. Check that the temporary pressed cursor clears and the expected hover/verb/system pointer takes over.

Also test a selected inventory item if your game uses item icons as cursors. That can explain why the pointer differs from the selected verb image.

GUI and display checks

Check the interface at its intended resolution and UI scale. Make the text box wide enough for realistic commands. Give your command label room for long object names and a two-object prefix.

Check a standalone build on each desktop platform you intend to ship. Desktop mouse cursors, touch input and gamepad/proximity interaction do not share exactly the same presentation. The shared commands should remain understandable on every device.

17. Fix common problems

What happens	What to check first
Wind exists but is missing from cycling or the built-in bar	Enabled on the verb; wind included in Global Settings > Cursor Verbs; then apply/save and restart the test.
Wind exists but no Wind button appears in my GUI	Add the button yourself. The custom GUI is not automatically rebuilt from Cursor Verbs.
Clicking Wind displays the winding message immediately	The response is on the button rather than the target. Use set-verb on the button and author Wind on the clock.
The button changes the word but the clock does nothing useful	Check the clock's enabled/clickable state, Wind Verb response, competing Object Events and current Interaction Mode.
A Context click keeps using Look	Check the target's Default Verb. Context does not use the same sequence as Verb Single Click.
The pointer stays normal over the toolbar or GUI	Move into the active room viewport. The editor/interface can restore its normal pointer.
The pointer stays normal in the room	Custom Cursor, verb-to-cursor link, cursor Enabled, usable image asset, and active cursor scope.
A hand cursor appears but the character does not pick anything up	A cursor is feedback. Add the actual gameplay response to the target or item.
Clicking feels offset from the visible tip	Check source-image hotspot X/Y, then the target's actual clickable area.
Cursor Scale makes a small image look larger	This build fits the image to a scaled reference size. See chapter 8.
Changing a cursor record seems to have no effect	Check object hover/press overrides, selected inventory icon and a forced cursor override.
Turning Custom Cursor off leaves an item icon	The selected-inventory-item cursor setting is separate.
A pressed cursor stays visible after release	Retest matching release and hover exit; check whether the apparent "pressed" cursor is actually the configured hover image.
The typed command finds no verb	Put the word in the correct verb's synonyms; avoid placing it only in an object's synonyms.
The typed command finds the wrong target	Check target names and synonyms for overlap; use a more distinctive alias.
Command Preview recognises the sentence but gameplay differs	Test the active room, target availability, authored actions and competing custom commands. Preview does not execute the interaction.

What happens	What to check first
Enter adds another line instead of submitting	Use a single-line Text Box; Multiline must be off for this parser-entry setup.
Enter appears to do nothing	Enable the parser, enable Run Text Parser Command on Enter, and test in the running game.
Use Lantern runs immediately rather than waiting for the clock	Allow Preposition on the Use verb, "on" entered, and Use Preposition opted in on the lantern.
A two-object typed command is rejected	Name both objects completely, use the configured preposition and check the source's opt-in/allowed-target response.
A renamed record breaks a selector or action	Restore or repair the old ID references. Use names for cosmetic wording changes.

Troubleshoot in a useful order

First check which command was selected. Then check which target was identified. Then check which response ran. Finally check the pointer's appearance.

That order prevents a visual symptom from hiding the actual input or action problem. If the parser and mouse reach the same target response but only one pointer looks wrong, concentrate on cursor selection rather than rewriting the puzzle.

18. A compact reference for your next project

Make another one-target verb

1. Add a unique verb ID and readable display name.
2. Add sensible synonyms.
3. Assign a cursor record, or deliberately use Default Cursor.
4. Include the verb in the normal offered set if required.
5. Author a response on each target that supports it.
6. Add or update your GUI controls.
7. Check both mouse and parser routes.

Make another cursor

1. Import compact, clear artwork with the transparency you need.
2. Create a cursor record and select that image asset.
3. Set the source-pixel hotspot.
4. Link the record from a verb or pointer-feedback field.
5. Test the actual rendered size and click alignment.
6. Check hover, press, release, inventory selection and GUI boundaries.

Keep these references distinct

Example reference	What it identifies
`wind`	The shared verb.
`hand_pointer`	A cursor record.
`pointer_hand`	The image asset used by that cursor.
`clock`	The target object.
`choose_wind`	The GUI button.
`clock_wound`	A game-state variable.

Use clear names, keep established IDs steady, and test the player's complete action from selection to response. Once that works, additional verbs become a repeatable piece of authoring rather than a new input system each time.