INTRODUCTION

Turn ordinary game text into part of the experience, from intimate dialogue and responsive HUD feedback to cinematic titles and world-space storytelling.

Start with one editable line, preview a visible result, then grow into reveal timing, sound, placement, reusable styles, and connected renderers only when your game needs them. Text Studio helps you succeed quickly while keeping ambitious treatments organized.

At a glance

Path	**What you can accomplish**
First success	Create a ready animated text object, write one line, and preview the result without entering Play Mode.
Effects	Explore 49 built-in effects visually, adjust readable controls, and apply them to a word, sentence, or selected range.
Reveals	Shape dialogue, subtitles, prompts, and long text by glyph, word, or line, with pauses, input, sound, and clear skip behavior.
Composition	Combine simple effects into a reusable treatment, then save successful timing and presentation choices for the rest of the game.
Placement	Keep normal text flow or arrange words on arcs, waves, circles, grids, and installed spatial forms.
Grow further	Connect UI Toolkit, 3D text, mechanical displays, terminal output, dialogue systems, localization, Timeline, and other optional tools when needed.

CONTENTS

WHAT YOU CAN CREATE

Text Studio turns ordinary game text into part of the experience. It can make dialogue feel spoken, rewards feel valuable, warnings feel urgent, title cards feel cinematic, and world-space labels feel alive, while the underlying text remains editable and accessible.

- Animate a single word, a complete sentence, or an entire long-text experience.
- Reveal by glyph, word, or line, with sound, pauses, input waits, markers, and skip behavior.
- Choose from 49 built-in effects, then combine them visually without writing code.
- Place text on arcs, waves, circles, grids, and extension-provided spatial layouts.
- Reuse a successful treatment through profiles, presets, composed effects, and Effect Code.

Rainbow shows how one readable word can become an immediate visual accent without changing the underlying text.

A good first goal: Create one sentence, preview one effect, then save the treatment only after it supports the moment in your game.

YOUR FIRST ANIMATED TEXT

REQUIREMENTS, INSTALLATION, AND VERIFICATION

Text Studio supports Unity 2022.3 or newer and uses TextMeshPro 3.0.9 or newer. Import Text Studio first, allow Unity to finish compilation, and import TMP Essential Resources when the project has not used TextMeshPro before. Optional integrations are imported only after their companion package is installed.

- Confirm that Tools > Text Studio contains Welcome, About, Gallery, Recipes, Health Check, and the setup commands.
- Create one world-space and one Canvas text object and confirm that both show a Text Studio inspector without missing-script warnings.
- Open Health Check and resolve missing TMP resources, duplicate event systems, unavailable renderers, or invalid project settings before authoring content.
- Keep the base package working before adding optional integration packages. A missing companion must never be required for ordinary Text Studio use.

Installation checkpoint: a newly created Animated TMP Text object should show readable text and a working inspector preview before you configure effects, profiles, or integrations.

ADOPT AN EXISTING TEXTMESHPRO OBJECT

Select an existing TextMeshPro or TextMeshProUGUI object and use its setup card to add Text Studio. The setup preserves the current text, font, layout, material, alignment, and Canvas or world-space role. After adoption, edit the authored line through Text Studio Raw mode. Text Studio remains the source owner during Live Preview and Play Mode so direct backend writes cannot silently discard markup.

1. Record the existing TMP text, overflow, font, and material settings when the object is production-critical.
2. Add Text Studio from the setup card and confirm that Raw contains the expected source.
3. Switch to Live, preview one effect, and check wrapping, alignment, material appearance, and layout bounds.
4. Update external scripts to call TextStudioText.SetText or TextStudioApi.ShowText instead of assigning TMP text directly.

STEP 1: CREATE THE TEXT

Choose GameObject > Text Studio > Animated TMP Text for world-space text, or Animated TMP Text (UI) for a Canvas. Text Studio creates a ready-to-edit TextMeshPro object and connects the animation workflow automatically. For an existing text object, use its setup card. Text Studio

selects an attached compatible backend such as UniText 3 automatically and falls back to TextMeshPro when needed. When Text Studio 3D is installed, eligible world-space objects also offer one-click 3D setup.

STEP 2: WRITE THE SOURCE

Open Content and enter the line your player should read. Use Raw when you want to edit the source. Use Live when you want to experience the resolved result.

```
You found a <shimmer>legendary reward</shimmer>!
```

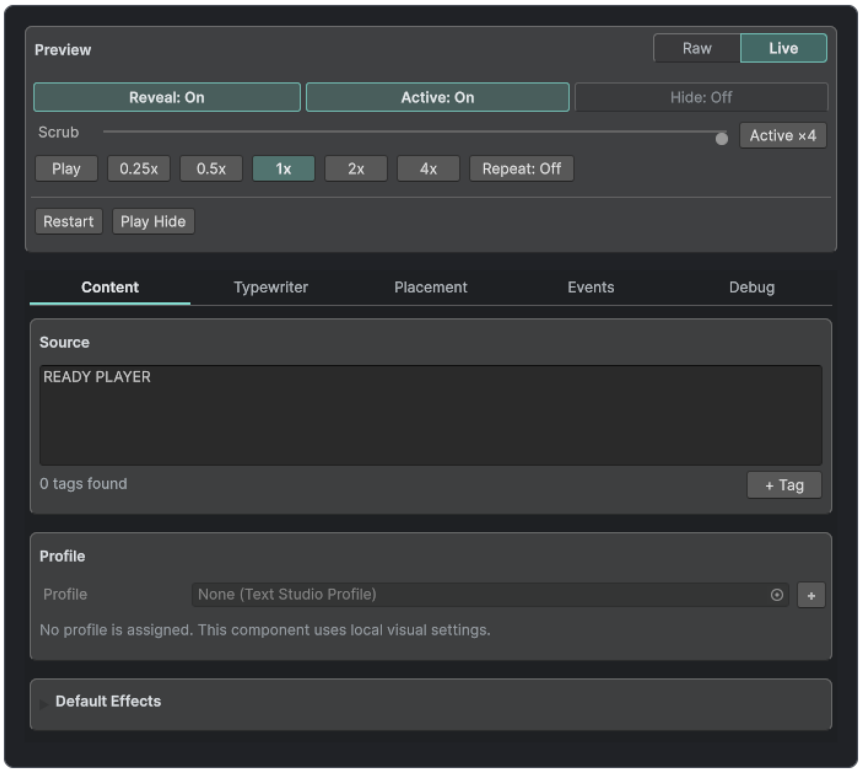
STEP 3: PREVIEW AND ADJUST

Press Play in the inspector preview. Start with the effect defaults, then adjust only the strength, speed, timing, or area that helps the line. The preview controls never rewrite your authoring settings.

STEP 4: CHOOSE HOW IT APPEARS

Open Typewriter. Choose Instant for a finished label, By Glyph for dialogue, By Word for subtitles, or By Line for staged instructions. Press Restart to experience the complete reveal again.

> **Success checkpoint:** You should now see the final text, a visible effect, and a repeatable reveal without entering Play Mode.



The main Text Studio inspector keeps source, typewriter, placement, events, debugging, and preview in one flow.

HOW TEXT STUDIO THINKS

SOURCE, LIVE PREVIEW, AND BACKENDS

Source is the text you author, including effect and action tags. Live Preview is the player-facing result after Text Studio resolves tags, profiles, external sources, and the active renderer. Edit in Raw and judge the experience in Live.

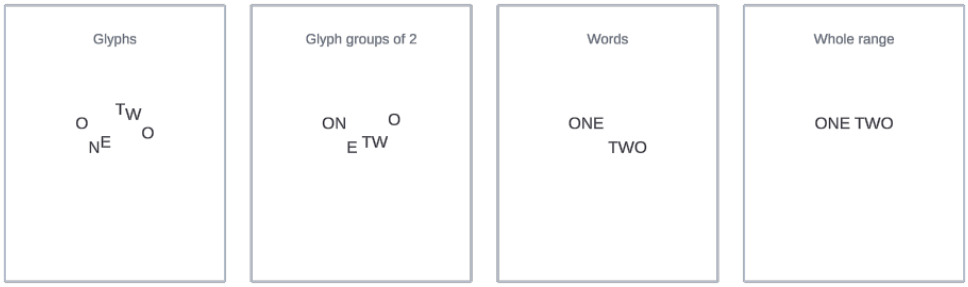
Text Studio automatically uses an attached compatible backend such as UniText 3 and falls back to TextMeshPro when needed. Raw mirrors direct component edits as authoring input. In Live Preview and Play Mode, direct backend writes warn and are rejected by default so rendered plain text cannot silently replace tagged source. The Expert Direct Backend Writes setting can deliberately Import or Ignore those writes; editor imports participate in Undo.

REVEAL, ACTIVE, AND HIDE

Reveal controls how text enters. Active controls what continues after the text is readable. Hide controls how it leaves. A treatment can use one phase or move smoothly through all three. Reveal-to-Active Blend prevents a hard visual jump when an entrance becomes a looping effect.

AREA AND SEQUENCE

An effect can act on each glyph, grouped glyphs, words, lines, paragraphs, authored ranges, or spatial fields. Sequence controls the order: forward, reverse, center-out, edges-in, all at once, or repeatable random.



The same Bounce effect grouped by glyphs, pairs, words, and the whole range.

MOTION POLICY

Project settings and each text object can choose Full, Reduced, or No Motion. Reduced Motion keeps the meaning while lowering displacement, rotation, flashing, or depth travel. No Motion keeps a stable readable result. Warnings appear when authored flashing can exceed three flashes per second.

READ THE INSPECTOR AS A WORKFLOW

The main inspector follows the order in which most text is built. Content owns the source and backend. Typewriter owns reveal and hide. Placement owns the glyph baseline. Events owns callbacks and authored beats. Debug explains the compiled result. Preview lets you experience those decisions together without entering Play Mode.

- Content: Raw and Live text, source ownership, profiles, default effects, and backend status.
- Typewriter: settings source, reveal unit, timing, motion, loop, skip, waits, and hide behavior.
- Placement: planar forms, extension providers, rotation, and spatial influence targets.
- Events: lifecycle callbacks, glyph callbacks, waits, markers, and relays.
- Debug: resolved text, parsed ranges, active effects, markers, ownership, timing, renderer capabilities, and current state.

Use Raw when changing authored markup and Live when judging what the player receives. Preview controls are temporary transport controls. They do not rewrite the serialized source, profile, typewriter recipe, or effect timing.

GLYPHS, GROUPS, RANGES, AND SEQUENCES

A glyph is one visible semantic text unit. It is not necessarily one UTF-16 character, because shaping, ligatures, surrogate pairs, combining marks, and fallback fonts can change how source characters become visible glyphs. Effects and reveal operate on visible glyph identity while Text Studio retains source mappings for diagnostics, markers, and integrations.

Area chooses the unit that receives one effect evaluation. Glyph treats every glyph independently. Word, Line, and Paragraph share one state across the corresponding semantic group. Range uses the authored tag range. Group Size combines consecutive units. Sequence then chooses the order in which those resolved units receive progress.

- Forward follows the source order. Reverse starts from the end.
- Center Out starts near the middle. Edges In starts at both ends.
- All At Once gives every unit the same progress.
- Seeded Random produces a stable shuffled order for the same source and seed.

THE THREE VISUAL PHASES

Reveal describes how a range enters, Active describes what continues after it becomes readable, and Hide describes how it leaves. A layer can participate in any combination. A reveal treatment should normally settle to the authored layout before an Active loop takes over, and a Hide treatment should begin from the current visible state.

Reveal-to-Active Blend crossfades the outgoing reveal contribution and incoming Active contribution for 0 to 5 seconds. Use a short blend when an entrance and a loop affect the same channel. Set it to zero only when an immediate handoff is intentional.

TIME, PLAYBACK, AND MOTION POLICY

Timing answers when a unit receives progress. A curve answers how that progress changes. Playback answers whether an effect runs once, loops, ping-pongs, follows an external weight, or holds a final value. These choices are independent of area and sequence.

Full Motion uses the authored treatment. Reduced Motion retains meaning while lowering travel, rotation, depth, flashing, or procedural instability. No Motion keeps a stable readable result. Author every production recipe with a restrained fallback, especially subtitles, warnings, and interactive UI.

> Accessibility rule: first reduce displacement, rotation, depth travel, flashing, and rapid color contrast. Do not remove timing, semantic emphasis, or readable state changes unless they are also unnecessary.

WRITING EFFECTS INTO TEXT

TAGS AND PARAMETERS

Wrap only the words that should be affected. Named parameters are easiest to review later; short aliases are useful when the line is already busy.

```
<wave amplitude=.18 speed=1.4>moving</wave>

<colorPulse from=#ff3355 to=#ffffff speed=1.7>critical</colorPulse>

<slide distance=.28>incoming</slide>
```

Appearance wrappers blend smoothly into Active by default. Set blend=0 for an immediate switch, or use blend=seconds from 0 to 5 when the reveal stack needs a deliberate handoff.

NESTING EFFECTS

Nest tags when two simple effects should act on the same words. Keep the combination purposeful and check readability at the smallest size used by the game.

```
<wave><colorPulse from=#42d9ff to=#d75cff>MAGIC</colorPulse></wave>
```

MARKUP SYNTAX CHOICES

Choose Compatibility (All Providers), Native (Text Studio Only), Text Animator, TMPEffects, or Custom Delimiters in Text Studio Project Settings. The Text Studio Text inspector deliberately follows that project-wide authoring policy instead of carrying a second per-object syntax switch. Compatibility is recommended for mixed content: it accepts native tags, the supported Text Animator brace set, and a safe TMPEffects subset without requiring either provider. Use the

migration preview to test representative lines before changing the project profile, then keep the chosen policy stable while content is in production.

OPEN, CLOSE, AND NEST TAGS

Canonical Text Studio markup uses angle tags. An opening effect tag starts a range, and a named closing tag ends that range. Tags are case-insensitive for lookup, but use the canonical catalog spelling so source remains easy to search and review.

```
<wave>Animated text</wave>

<wave amplitude=.18 frequency=.65 speed=1.2>Animated text</wave>
```

The anonymous closing tag </> closes the most recently opened compatible effect or native rich-text range. It does not close every open range at once. Repeat it to unwind a nested stack. Named closing tags are clearer in shared dialogue files; anonymous closings are useful for compact generated or migrated markup.

```
<wave><color=#66CCFF>Nested text</></>

<i>A<wave>B</>C</>
```

An unclosed Text Studio effect continues to the end of the source. This is useful for a treatment that intentionally covers the remainder of a line, but diagnostics should be reviewed because an accidental missing close can affect more text than expected. Mismatched named closings remain visible in diagnostics.

COMPATIBILITY PROFILES AND PROVIDER SYNTAX

Compatibility (All Providers) accepts canonical Text Studio angle tags, a focused Text Animator brace set, and the safe TMPEffects subset. Native isolates Text Studio markup. Text Animator and TMPEffects each isolate one provider dialect, which is useful for deterministic migration and for content owned by one external authoring format.

```
{fade}Reveal this{/fade}

{#fade}Hide this{/#fade}

<wave>{fade}{#incr}Compatible range{/#incr}{/fade}</wave>

<!wait=0.25><jump amp=8>TMPEffects source</>
```

Known Text Animator braces preserve supported timing parameters. Supported TMPEffects animations, show and hide ranges, writer commands, and events translate into canonical Text Studio markup. Unknown or unsupported provider tags remain visible and produce source-ranged diagnostics instead of executing silently.

> Provider compatibility is dependency-free and intentionally focused. It translates a reviewed syntax subset; it does not load external provider assets, components, databases, APIs, or arbitrary commands.

PARAMETERS, ALIASES, AND VALUES

A tag can use positional parameters, named parameters, or supported short aliases. Named parameters are easiest to maintain because their meaning remains visible. Parameters are

resolved in source order, and the last valid occurrence wins when the same setting appears more than once.

```
<wave amplitude=.18 frequency=.65 speed=1.2>Named values</wave>
<wave a=.18 f=.65>Short aliases</wave>
<colorPulse from=#FF3355 to=#FFFFFF speed=1.7>Critical</colorPulse>
```

Use speed for a rate and duration for a time span. Playback values include infinite or persistent, loop, pingpong, once or short, and external or weighted where the effect supports them. Colors accept the formats described by the tag browser. Vectors use comma-separated components. Boolean values should be written as true or false.

TEXTMESHPRO RICH TEXT INSIDE TEXT STUDIO

Text Studio consumes its own effects, actions, expansions, and extension tags while preserving supported TMP rich text for the TMP backend. You can combine bold, italic, underline, strikethrough, color, alpha, size, case, mark, subscript, superscript, font, weight, gradient, link, style, spacing, alignment, margins, line height, position, rotation, material, scale, line breaks, pages, spaces, sprites, and soft hyphens with Text Studio ranges.

```
<wave><b><color=#66CCFF>Important</color></b></wave>
```

Native formatting is not an animated effect binding. It remains in Display Text so the backend can render it. Anonymous </> closings are translated to the correct named TMP close. Use <noparse> for backend-supported literal markup regions, or escape angle characters with a backslash when the source should display them.

```
\<wave\>This is literal markup\</wave\>
```

CUSTOM DELIMITERS AND SYNTAX ASSETS

Leave Markup Syntax unassigned for the recommended Compatibility (All Providers) profile. Native accepts Text Studio angle tags only. Text Animator and TMPEffects isolate their supported provider dialects and keep unrelated authoring tags literal. Custom Delimiters converts one chosen opening and closing character pair into Text Studio angle markup and can optionally continue accepting native angle tags. The opening and closing characters must be different and a custom tag must remain on one line.

1. Create Assets > Create > Text Studio > Markup Syntax.
2. Choose Custom Delimiters and enter one opening and one closing character, such as [and].
3. Decide whether authors may continue using native angle tags.
4. Assign the syntax asset in Text Studio Project Settings and test representative source in Raw and Live modes.

```
[wave amplitude=.12]Custom syntax[/wave]
```

TAG EXPANSION SHEETS

A Tag Expansion Sheet gives writers one short project tag that expands into a reviewed opening stack. Auto Closing derives the required named closings in reverse order. Custom Closing is available when the opening markup contains a project-specific construct that automatic derivation cannot represent.

1. Create Assets > Create > Text Studio > Tag Expansion Sheet.
2. Add an entry with Tag ID warning.
3. Set Opening Markup to <shake strength=.08><colorPulse from=#FF3344 to=#FFFFFF><emphasis>.
4. Keep Closing Mode on Auto and confirm the preview derives </emphasis></colorPulse></shake>.
5. Register the sheet in Text Studio Project Settings, then use <warning>Reactor unstable</warning>.

Project sheets load as the stable catalog layer. Runtime registration can add or override expansions for generated content. Duplicate IDs and invalid closing stacks appear in diagnostics. Effect Code can bundle required expansion dependencies when a composed treatment is shared.

ESCAPING, DIAGNOSTICS, AND AUTHORING HELP

Use the Content tab's tag browser and inline suggestions instead of memorizing every ID. Search by effect name, alias, category, parameter, or integration. The Debug tab lists parsed occurrences with source ranges, resolved text, effect bindings, action markers, implicit end-of-source ranges, and unmatched tags.

- If markup appears on screen, confirm that the tag is known and the current authoring profile accepts its syntax.
- If too much text animates, inspect the matching close or the implicit end-of-source range.
- If a tag works in TMP but not UI Toolkit or 3D, inspect the renderer-channel compatibility warning.
- If a migrated brace remains literal, confirm that it belongs to the supported compatibility set.

TAG CATEGORY REFERENCE

The appendix provides the complete visual effect gallery. Use these additional categories to understand how non-effect markup participates in the line.

- Actions: <wait>, <pause>, <resume>, <speed>, <speedUp>, <speedDown>, <waitForInput>, and <event>. Actions are point-in-time instructions and do not require a closing tag.
- Appearance and disappearance wrappers: <appear effectId> and <disappear effectId> assign registered effects to a lifecycle phase.

- Character substitution: <scramble> changes presentation glyphs while preserving authored text, layout identity, marker positions, and source mappings.
- Decorations and TMP formatting: supported native tags remain in Display Text for the active backend.
- Expansion tags: project-defined short tags expand into reviewed Text Studio stacks.
- Extension tags: optional packages can register package-owned syntax and processors without adding a hard dependency to the base package.

For every effect tag, the appendix now lists the canonical ID, practical purpose, starting example, visual variations, and source-verified key controls. The website reference can expand those entries into individual searchable pages with aliases, units, complete parameters, backend support, and animated examples.

BUILD REVEALS AND TYPEWRITER EXPERIENCES

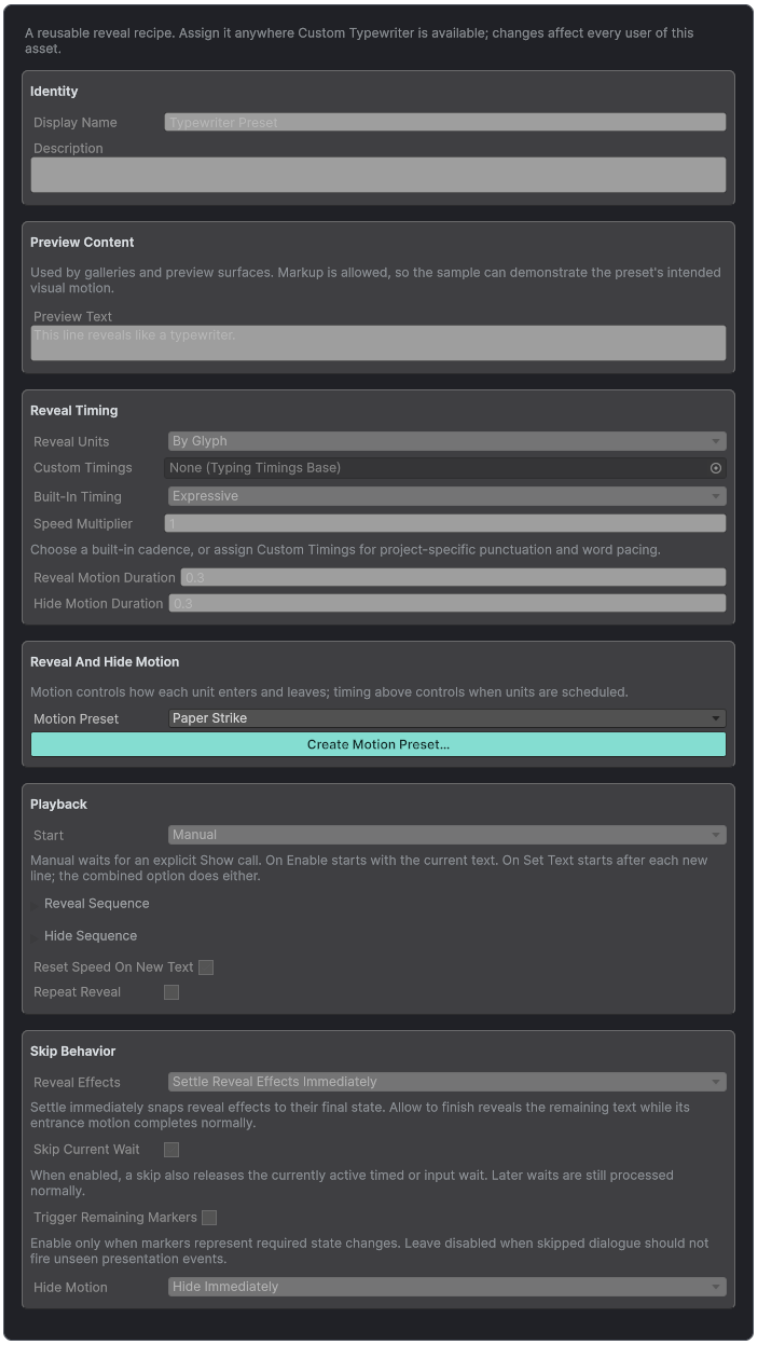
CHOOSE A SOURCE

Typewriter can be None, Built-In, Custom, or Inline. Built-In is the fastest start. Custom points to a reusable preset. Inline exposes the complete settings on this text object. A built-in or custom choice can be converted to Inline when this one line needs a local variation.

SHAPE THE PACE

- Reveal by glyph, word, or line.
- Use a timing preset or assign custom character and word timings.
- Choose forward, reverse, center-out, edges-in, all-at-once, or seeded-random order.
- Add reveal and hide motion without changing the content.
- Choose start, loop, delay, skip, and input behavior deliberately.
- Use punctuation-aware audio and marker cues for dialogue rhythm.

Runtime dialogue code can check IsWaitingForAction before calling SkipCurrentWait(). This distinguishes a timed or input action wait from an ordinary paused reveal.



A reusable Typewriter Preset keeps pacing, motion, playback, and skip behavior consistent.

REVEAL UNITS AND SETTINGS SOURCES

Instant displays the complete line. By Glyph gives the familiar character-by-character result. By Word is often better for fast subtitles because it preserves reading rhythm without creating a machine-gun stream of letters. By Line stages instructions or credits. Region and Custom modes are available for specialized providers.

None disables typewriter behavior. Built-In selects a curated recipe. Custom assigns a reusable Typewriter Preset. Inline exposes the complete local recipe. Convert a built-in or custom choice to Inline when one object needs a deliberate exception. Assign a Profile when several objects should share the same complete defaults.

TIMING PRESETS

- Uniform: predictable mechanical pacing.
- Fast: responsive HUD, notifications, and short interactions.
- Slow: deliberate reveals with room for voice or atmosphere.
- Expressive: stronger punctuation rhythm for dialogue.
- Cinematic: longer beats for titles and staged presentation.
- Staccato: clipped rhythm for alerts, terminals, or stylized speech.
- By Word: word-level pacing without extremely fast glyph ticks.
- Typist: familiar keyboard-like timing variation.
- Narrative: readable prose pacing with punctuation-aware pauses.

CHARACTER AND WORD TIMINGS

Character Timings use full text context for abbreviations, punctuation runs, neighboring units, words, lines, paragraphs, inline instant ranges, newline handling, and final-character behavior. The first chosen unit appears immediately; punctuation waits begin after the mark is visible; adjacent punctuation can collapse into one deliberate beat. Word and Line modes apply one boundary delay per group, reordered sequences retain each unit's authored pacing, and the configured trailing wait remains part of reveal completion. Word Timings control word and punctuation-boundary delays. Built-in cadences and custom timing assets expose Wait After Final Reveal to control whether completion includes the final trailing delay. Use <instant>...</instant> to reveal a marked span atomically.

```
Dialogue test: "Dr. Vale... wait!"
```

Test abbreviations and ellipses together: Dr. should not receive a sentence-ending pause, while an ellipsis should still create one deliberate beat. Verify the result in the timing preview and Debug summary before adding explicit wait tags.

CLOCK AND UPDATE BEHAVIOR

Choose scaled time for gameplay-bound presentation and unscaled time for menus, pause screens, or dialogue that must continue while Time.timeScale is zero. Update phase determines when Text Studio advances relative to gameplay and UI. Maximum Delta prevents a long suspended frame from instantly consuming an entire reveal. Edit Mode ticking exists for preview, not as a substitute for runtime testing.

REVEAL AND HIDE MOTION

Reveal timing decides which unit is current. Reveal Motion decides how that unit enters. Hide Motion independently decides how it leaves. Use None for a stable reveal, a built-in motion for quick setup, a reusable motion preset for project consistency, or a custom tag stack for a specific treatment.

Reveal Appearance Duration and Disappearance Duration control the visual transition around the timing point. Sequence settings can run forward, reverse, center-out, edges-in, all at once, or seeded random. Loop Reveal restarts the complete experience after its delay. Start behavior decides whether reveal begins automatically or waits for an explicit call.

SKIP, CONTINUE, WAITS, AND HIDE POLICY

SkipReveal completes the current line according to the configured settle policy. AdvanceOrSkip is intended for continue buttons: it first completes an active wait or reveal, then returns true only when the owning dialogue system may advance. SkipCurrentWait cancels a cooperative timed, input, or audio wait without confusing it with an ordinary paused reveal.

```
public void ContinuePressed()
{
    if (dialogueSource.AdvanceOrSkip())
        ShowNextLine();
}
```

Choose whether skipping triggers remaining markers, skips the current wait, settles reveal motion, or jumps to the final visible state. Hide can play its configured sequence, be skipped to its end, or be stopped. Test rapid repeated input, non-skippable custom actions, and a continue press during the first frame of a new line.

CREATE COMPOSED EFFECTS

Use a Composed Effect when a treatment needs several coordinated layers, reusable timing, spatial influence, or a designer-friendly asset. Choose a 2D or 3D authoring space first. Text Studio hides unavailable choices but preserves existing 3D data if an add-on is temporarily absent.

A SMALL, SUCCESSFUL RECIPE

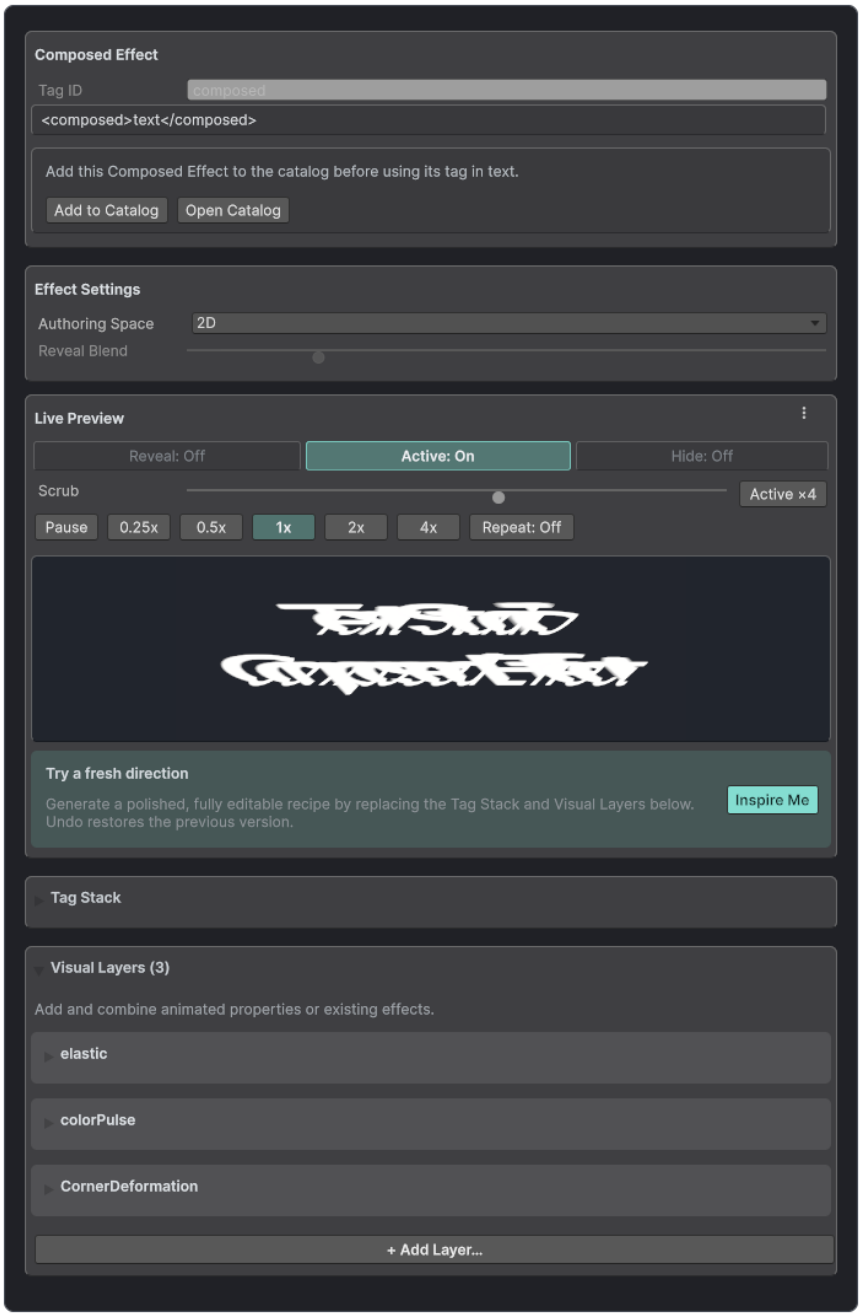
- Add Elastic for the reveal.
- Add Color Pulse for the readable active state.
- Add Corner Deformation only if the silhouette benefits from it.
- Choose phases, sequence, area, curve, and influence for each layer.
- Solo a layer while tuning it, then compare the complete result.
- Use Extract when one useful layer should become a reusable asset.

VISUAL LAYERS AND NESTED EFFECTS

Visual layers cover position, rotation, scale, alpha, color, skew, UV movement, pivot, corner deformation, and extension channels. Each layer can also select original text colors by hue, saturation, perceptual luminance, or alpha, with smooth tolerance and inversion controls. Nested effect layers reuse any registered effect with typed controls, area, phase, timing, scope, and staggering.

PREVIEW AND SHARE

Switch between front, perspective, side, and orbit views. Use Draft while iterating and Full for final judgment. Compare keeps two states visible. Effect Code copies a complete recipe with readable metadata and bundled dependencies, then imports it with conflict-safe identities.



The Composer combines effect layers, visual channels, timing, scope, and a live result.

THE COMPOSER WORKSPACE

Identity defines the asset name, catalog information, and optional Tag ID. Authoring Space declares whether the asset is planar or can use installed 3D channels. The usage snippet shows the exact opening and closing markup. Effect Settings control the global reveal-to-active handoff. Live Preview owns view, phase, quality, playback, scrubbing, comparison, and export.

- Layer Stack is the ordered evaluation list.
- Add Layer opens built-in effects, custom effects, composed effects, and typed visual channels.
- Inspire Me creates a reviewable starting stack, not hidden runtime behavior.
- Effect Code copies the treatment and its required dependencies with conflict-safe identities.
- Draft preview prioritizes iteration speed. Full preview is the final visual check.

LAYER OPERATIONS AND EVALUATION ORDER

Enabled controls whether a layer participates in the result. Solo is preview-only and isolates one layer without changing the serialized stack. Rename layers by intent, such as Reveal Lift or Active Highlight. Reorder by dragging. Duplicate before experimenting. Delete only after checking nested dependencies. Extract creates a reusable nested Composed Effect when a useful sub-stack should be shared.

Layers evaluate from top to bottom in the displayed order, and each layer sees the state produced by earlier layers. This matters for Override and Lerp behavior, source versus accumulated color, and two layers writing the same transform channel. Keep independent channels separate and place deliberate overrides after the contribution they replace.

VISUAL CHANNELS

- Position adds or blends a local glyph offset.
- Rotation composes a per-glyph quaternion around the current pivot.
- Scale multiplies or blends the current glyph scale.
- Alpha multiplies or interpolates visibility without changing source text.
- Color can set, multiply, or interpolate the current glyph color.
- Skew changes the glyph quad without changing text layout.
- UV Offset moves material sampling where the active renderer supports it.
- Pivot changes the origin used by rotation and scale.
- Corner Deformation offsets individual glyph corners for flags, ripples, squeezes, and custom silhouettes.
- Registered extension channels retain stable namespaced IDs so optional renderers and add-ons can add capabilities without changing the base asset format.

A layer should change only the channels it intends to own. Position must not silently reset color, and a color layer must not replace reveal or placement. Unsupported channels are preserved in the asset and reported by diagnostics instead of being silently discarded.

PHASES, CURVES, PLAYBACK, AND PHASE SHIFT

Assign Reveal, Active, and Hide independently on every layer. The Curve shapes normalized progress. Playback decides how that curve is repeated or externally driven. Duration and speed set the layer clock. Phase Shift offsets the layer relative to the resolved unit index, and timing stagger offsets units without changing their area.

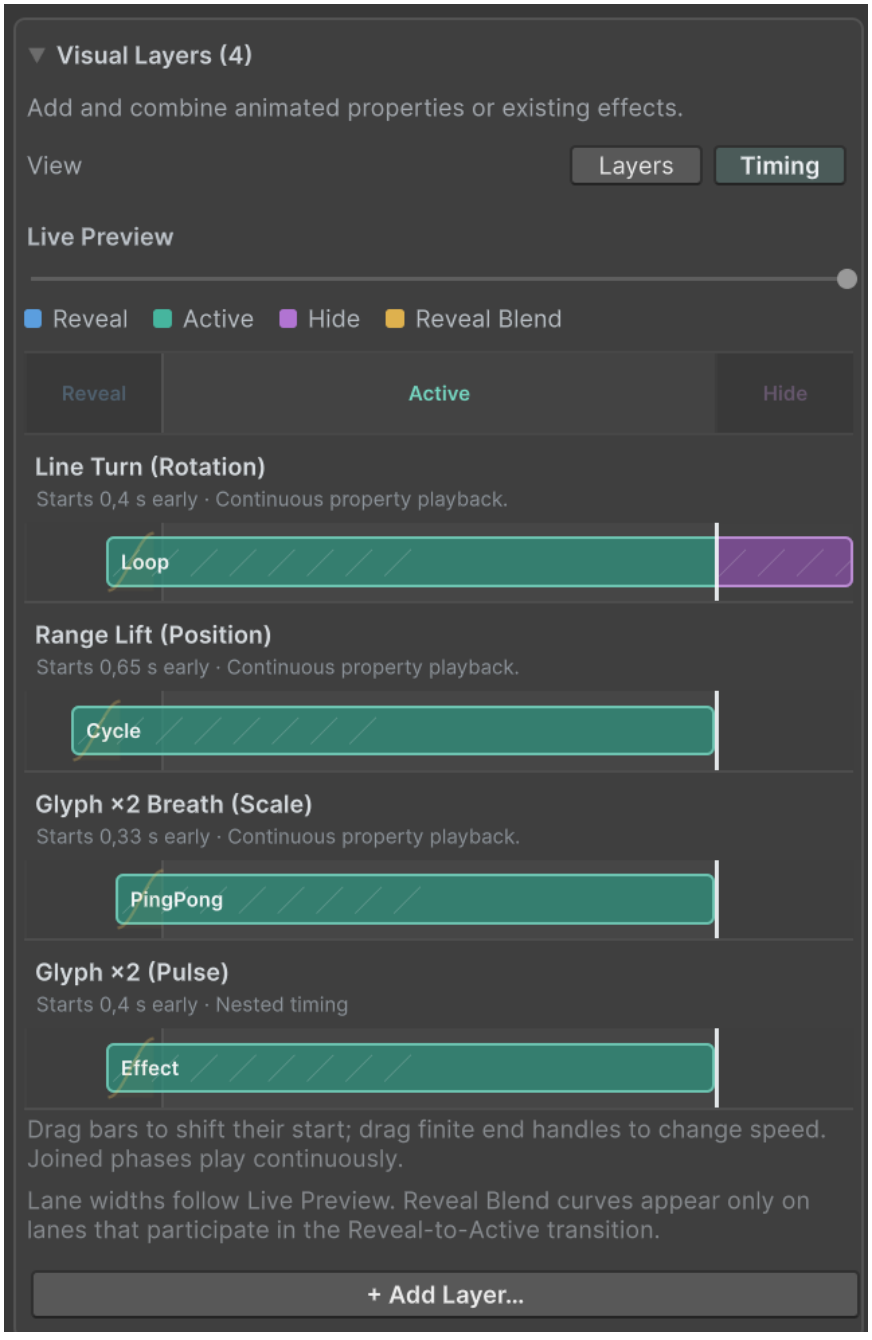
Once runs a finite treatment. Loop repeats. Ping-Pong alternates forward and return. Weighted follows an input curve. External follows supplied progress. Start delay, forward time, peak hold, return time, cycle count, cycle gap, and completion behavior belong to the playback recipe.

VISUAL TIMING VIEW

Switch Visual Layers from Layers to Timing to compare every layer against the same Reveal, Active, and Hide lifecycle used by Live Preview. Scrub the shared playhead, drag a bar to shift its start, and drag a finite end handle to change playback speed. Continuous and nested lanes remain visibly distinct so timing can be reviewed without opening each layer card.

Adjacent enabled phases join without gaps or playback restarts. Reveal Blend appears only on lanes that participate in the Reveal-to-Active transition, keeps its full authored duration when it

crosses into Active, follows layers shifted into Reveal, and does not fade in a layer that begins later inside Active.



The Timing view aligns every visual layer to Live Preview and exposes phase shifts, finite playback length, joined phases, nested timing, and Reveal Blend.

Reveal, Active, and Hide preview choices are remembered for the project when you move between Composed Effects. Keep one comparison view active while tuning a family of dialogue, reward, or warning recipes, then switch phases deliberately when the design question changes.

AREA, GROUPING, AND SEQUENCE PER LAYER

Each layer can target Glyph, Word, Line, Paragraph, or Range and can combine consecutive units with Group Size. Forward, Reverse, Center Out, Edges In, All At Once, and Seeded Random then determine progression. A nested effect can keep its own default area, but the parent layer may deliberately override it for the composed treatment.

```
Example: Area = Word, Group Size = 1, Sequence = Center Out
```

BLEND MODES AND SOURCE COLOR SELECTION

Additive contributions accumulate with the current channel. Multiply scales it. Override replaces it. Lerp interpolates between the current and requested value. For color, Set applies a new color, Multiply preserves underlying variation, and Alpha Only changes transparency without repainting the glyph.

Source Color selectors evaluate the immutable original glyph color before composed color layers accumulate. Select by Hue, Saturation, Perceptual Luminance, or Alpha, then tune tolerance, softness, and inversion. This makes it possible to animate only the bright part of a gradient title or a package-authored color region while preserving the rest.

SPATIAL INFLUENCE FIELDS

Influence fields multiply or combine with the semantic Area result. Linear creates a directional ramp. Sphere and Box create bounded volumes. Cylinder and Torus create radial forms. Target Distance follows the shared Spatial Effect Target. Noise adds repeatable variation.

- Source Layout uses the original local text layout.
- Source Text Normalized uses a stable 0 to 1 rectangle across the original visible text.
- Placed Text evaluates after the active placement has moved glyphs.
- World evaluates the placed glyph center in world space.

Choose X, Y, or Z axes where applicable. Use Multiply to gate an existing influence, Add to accumulate, Maximum to keep the strongest field, or Minimum to keep the weakest. Strength, falloff, inversion, and deterministic noise controls let the stack remain reproducible.

SURFACE MASKS AND 2D OR 3D AUTHORING

A surface mask is meaningful only when the active renderer exposes semantic surfaces. TMP and UI Toolkit can still use transform, color, alpha, and compatible material channels. Text Studio 3D adds Front, Front Relief, Front Bevel, Sides, Inner Sides, Back Bevel, Back, Edges, and Decorations classifications.

Choose a 2D authoring space for renderer-neutral effects. Choose 3D when the treatment intentionally uses extrusion, contour, bevel, emission, metallic, smoothness, dissolve, or other

registered 3D channels. If the add-on is absent, Text Studio preserves the 3D data and reports unavailable capabilities.

When a 2D Composed Effect contains a hidden Z position, X or Y rotation, or Z scale value, the layer now explains the exact incompatible value. Use Fix for 2D to reset only that hidden value, or switch Authoring Space to 3D when the depth was intentional.

SEVEN PRODUCTION RECIPES

1. Reward pop: Reveal scale overshoot, short position lift, warm color Set, and a brief shimmer. Add a Reduced Motion version with color and alpha only.
2. Dialogue emphasis: Word area, restrained Active pulse, Multiply color, and a short finite playback so the sentence becomes still.
3. Warning signal: Range area, red color sweep, low-amplitude shake, and an event marker at the warning beat. Cap flashing frequency.
4. Karaoke highlight: External playback drives a color sweep by progress. Use Source Color selection to preserve speaker styling and add a short release tail.
5. Hologram title: Chromatic shift, alpha scan, UV motion, and an optional 3D surface mask. Keep text readable when material channels are unavailable.
6. Four-corner deformation: Corner offsets, an intentional pivot, forward sequence, and bounded strength so counters and small text remain legible.
7. Hide treatment: Fold or slide, alpha fade, reverse order, and a skip policy that settles to fully hidden without leaving an active layer.

SHARING AND DIAGNOSTICS

Copy Effect Code when a treatment must cross projects or packages. The payload records readable metadata, layers, stable IDs, and required dependencies. Import reports identity conflicts and preserves unavailable extension channels. Use Extract when reuse should remain inside the project as a normal asset reference.

- A layer appears late: inspect start delay, phase assignment, playback, phase shift, and sequence.
- A layer never appears: inspect Enabled, Solo state, Area range, influence, source-color selector, surface mask, and renderer support.
- The result pops after reveal: increase Reveal-to-Active Blend or align the two layers' final values.
- Two layers fight: inspect their evaluation order and whether the later layer uses Override or Lerp.
- The composition is expensive: isolate topology, material, and high-frequency layers, then reduce update frequency or move stable work into a preset.

PLACE AND INFLUENCE TEXT

PLACEMENT SHAPES THE BASELINE

Linear keeps the normal TMP layout. Arc, Wave, Circle, and Grid reposition glyphs without rewriting the source. Placement extensions can add specialized choices such as the spatial forms supplied by Text Studio 3D.

SPATIAL FIELDS SHAPE EFFECT STRENGTH

Linear, Sphere, Box, Cylinder, Torus, Target Distance, and deterministic Noise fields can be stacked. Evaluate them in Source Layout, Placed Text, or World coordinates. Normalized source-text coordinates remain stable even when placement, animation, object transforms, or world movement change. Assign a Spatial Effect Target when motion should react to a player, cursor, point of interest, or moving object.

> **Keep the distinction clear:** Placement decides where letters live. An effect decides how they change over time. A spatial field decides where that effect is strongest.

PLACEMENT MODES

Linear preserves the backend layout. Arc bends the baseline by radius and total angle. Wave displaces the baseline through a repeatable curve. Circle wraps text around a full or partial ring. Grid assigns glyphs to rows and columns. Extension providers add package-owned planar or spatial arrangements.

Rotation With Placement can preserve original glyph orientation or follow the placement tangent. Test direction and alignment with multiline text, mixed case, punctuation, and narrow layout bounds. Placement changes glyph origins but does not rewrite source text, TMP line wrapping, or effect tags.

UNITY SPLINES PLACEMENT

Install Unity Splines 2.0.0 or newer, then import the Text Studio Unity Splines integration. Add SplinePlacementProvider, assign a SplineContainer and spline index, and select the provider through the ordinary Placement workflow. Text Studio continues to own source, effects, timing, selection, and dispatch. The provider owns only sampled positions and orientation.

SPATIAL EFFECT TARGET

Assign a Transform when Target Distance fields or interactive effects should follow a player, cursor, controller, camera point, or gameplay object. Text Studio snapshots the target in local and world space and invalidates visual state when the owner or target moves. Keep target motion separate from baseline placement so the same field remains reusable.

SOUND, INPUT, EVENTS, AND INTERACTION

ACTIONS INSIDE THE LINE

Wait, speed, input, event, marker, and sound actions let authored text control pacing without splitting the sentence across scripts. A Play Sound action can use an asset or component, stop on skip, and optionally wait for completion with a timeout.

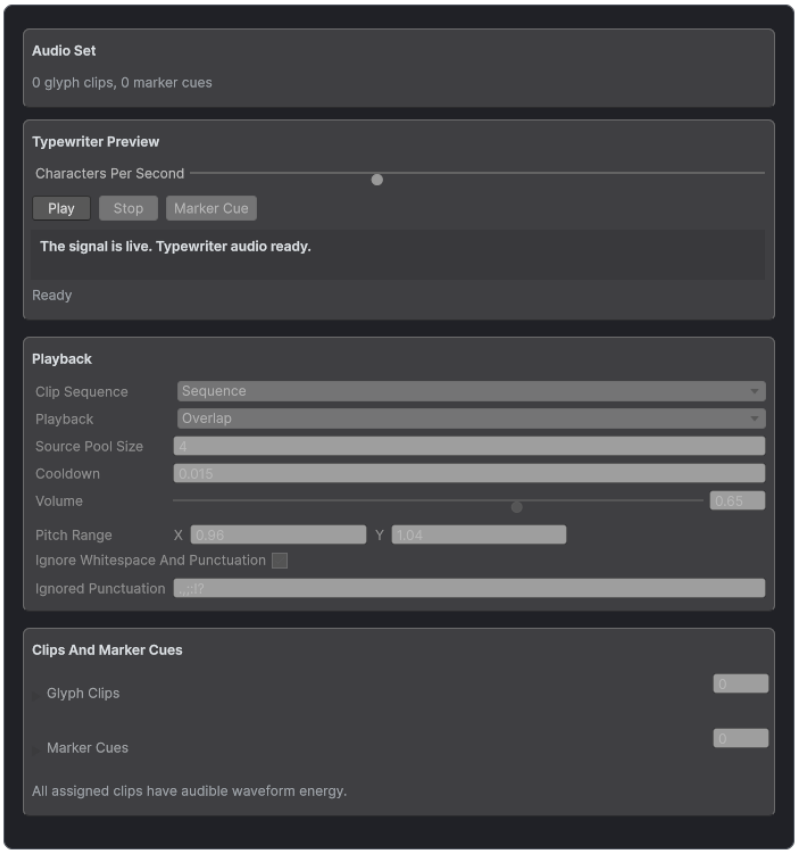
```
First beat<wait seconds=.35> waits. <speed x=2>Then the pace changes.</speed>
```

Open the tag browser for a guided setup when an action has parameters. Required values stay enabled, optional values can be included deliberately, and Select offers compatible routes or

identifiers discovered from the current text. Generated Markup is type-checked before Copy or Insert, and browsing never invokes runtime-only callbacks.

AUDIO SETS AND PLAYERS

An Audio Set defines the available clips and marker cues. The Audio Player decides how glyphs, words, punctuation, and markers trigger them. Keep frequent glyph sounds short and varied enough to avoid fatigue.



Audio Sets preview glyph clips and marker cues before they are connected to dialogue.

BUILT-IN ACTION TAGS

- <wait=0.5> pauses reveal for a cooperative timed wait. Named form: <wait seconds=0.5>.
- <pause> pauses indefinitely until code or a later action resumes reveal.
- <resume> releases an ordinary pause.
- <speed=1.25> sets the reveal multiplier. <speedUp=2> and <speedDown=2> change it relative to the current pacing.
- <waitForInput input=submit timeout=5> waits for a keyed or any-input provider and may time out.
- <event=doorOpen,cellar> raises a named route with optional payload values.
- <playSound> invokes a registered Play Sound asset or component and can optionally wait, time out, and stop on skip.

Actions are point-in-time markers and do not wrap visible text. They run when reveal reaches their compiled position. Self-closing syntax is accepted, but the short non-closing form is common because the action itself has no range.

The searchable tag browser keeps its catalog compact. Select an action that needs parameters, then choose Configure to open the guided setup only when it is useful. The setup validates required values, offers contextual route choices, and shows the exact markup that will be inserted before it changes the source.

> Authoring pattern: search by intent, preview the exact action markup, and insert only after the route and parameters are valid. This is especially useful for sound cues, routed events, input waits, and project-specific actions.

INPUT SYSTEM WAITS

The built-in waitForInput action falls back to the legacy any-advance provider. Install Input System 1.4.0 or newer to use InputActionReference mappings. Create named mappings such as submit, cancel, or next, decide whether each action should auto-enable, and use the same key in markup. A timeout greater than zero resumes automatically; zero or less waits until input or a cooperative skip.

AUDIO SETS AND AUDIO PLAYER

An Audio Set owns reusable glyph clips and named marker cues. The Audio Player decides when to play them, how to choose clips, whether sounds overlap or interrupt, the cooldown, punctuation policy, pitch and volume variation, and the AudioSource pool. Preview assets before placing them in a scene, then test a fast reveal and a full skip so multiple glyphs revealed in one frame do not overwhelm the mix.

Play Sound action assets and components are for authored one-shot cues that may wait for completion. Glyph audio is for the continuing texture of a typewriter. Marker cues are for semantic beats. Use the smallest mechanism that owns the desired timing.

EVENTS, MARKERS, AND RELAYS

The Events tab exposes typewriter start, shown, disappeared, glyph-revealed, wait-started, wait-finished, and marker messages. Use TextStudioMarkerEventRouter to map named event routes to UnityEvents and choose whether a binding receives the route, first parameter, or joined payload.

```
The door opens.<event=doorOpen,cellar> Stay close.<event=music,tense>
```

Use TextStudioTypewriterEventRelay when designers need inspector-wired callbacks without a custom subscriber. Per-glyph events can fire frequently, so keep expensive work, allocations, and scene searches out of those listeners.

DIALOGUE SOURCE AND LONG TEXT

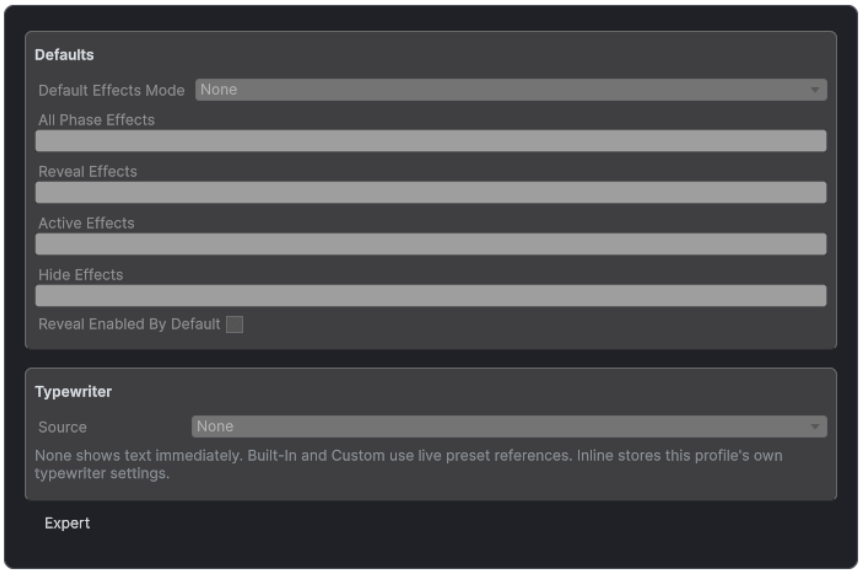
TextStudioDialogueSource is the stable bridge for dialogue, quest, subtitle, or cutscene systems. ShowLine stores the line, updates the hub, and optionally restarts reveal. AdvanceOrSkip completes the current wait or reveal before asking the owning system for another line.

TextStudioPageNavigator works with TMP Page overflow and can follow the revealed glyph without changing reveal progress. TextStudioAutoScrollFollower keeps logs, subtitle history, tutorials, and credits visible inside a ScrollRect. Test page changes after localization, font fallback, and narrow layout changes.

REUSE YOUR WORK

PROFILES

A Profile is the best home for defaults shared by many text objects, including default effects, typewriter behavior, and how direct TMP writes are handled. Assign a profile to a prefab or scene object, then keep only meaningful local overrides.



Profiles collect the defaults that should stay consistent across a game or feature.

PROJECT AND SHARED SETTINGS

Project Settings control global defaults, markup syntax, the effect catalog, and motion policy. Shared Settings are useful when several objects should point to one reusable configuration. Clear an assigned profile whenever you need to return to complete inline control.

WHAT YOU CAN CREATE

- Profiles and Shared Settings
- Typewriter Presets and Typewriter Motion Presets
- Character Timings and Word Timings
- Markup Syntax and Tag Expansion Sheets
- Audio Sets, sound actions, and input actions
- Effect Curves, including Linear, Bounce, Step, Hold, and Custom, plus Once, Loop, finite Ping-Pong, and Weighted playback assets

CHOOSE THE RIGHT REUSABLE ASSET

- Profile: complete defaults shared by many Text Studio objects.
- Typewriter Preset: reveal, timing, motion, loop, wait, skip, and hide behavior.
- Typewriter Motion Preset: reusable Reveal or Hide motion stack.
- Character Timings or Word Timings: pacing rules only.
- Composed Effect: ordered reusable visual treatment.
- Effect Curve: reusable progress shape.
- Effect Playback: reusable temporal envelope.
- Markup Syntax: project authoring delimiter policy.
- Tag Expansion Sheet: project vocabulary that expands into reviewed markup.
- Audio Set: reusable glyph clips and marker cues.

Use shared assets for decisions that should remain consistent. Use Inline settings for a deliberate local exception. When a Profile is assigned it temporarily supplies profile-owned defaults, but Text Studio retains the full local configuration and restores it when the profile is cleared.

PROJECT SETTINGS AND CATALOG OWNERSHIP

Text Studio Project Settings define global defaults, the active syntax policy, project effect and action registrations, tag expansion sheets, motion policy, and runtime loading behavior. Runtime registrations have the highest priority, project entries remain available underneath, and built-ins remain the final fallback.

EFFECT CODE

Effect Code is for portable treatments. Copy from the composer, review the metadata and required dependencies, then import into another project. Conflicting identities are resolved without silently overwriting unrelated assets. Keep package-owned extension IDs namespaced and preserve missing dependencies until their add-on is installed.

COMMON GAME SCENARIOS

DIALOGUE AND SUBTITLES

Start with By Glyph or By Word, punctuation-aware timing, gentle reveal motion, skippable waits, and short audio cues. Use a marker router for speaker, portrait, camera, and choice events. Reduced Motion should keep timing and meaning without unnecessary travel.

TEMPORARY TEXT AND GAMEPLAY FEEDBACK

Use temporary text when each request is a short-lived event: damage over an enemy, a heal near the player, a status word above a target, a combo tier on the HUD, or a score burst near a reward. TextStudioEmitter owns the reusable instance lifetime while TextStudioText, profiles, effects, motion policy, and the chosen renderer keep the same responsibilities they have for persistent text.

Start from GameObject > Text Studio > Text Emitter. Use the compact + actions to create a complete host and an Emission Preset, or assign the ready-to-customize Damage, Critical, Heal, Status, Combo, and Score assets.

Choose the host by presentation, not by gameplay meaning: Canvas for HUD feedback, world TextMeshPro for in-scene labels, or a complete Text Studio 3D host for extruded feedback. The emitter and preset API stay the same.

Let the preset own the profile, numeric template, shared number format, Active duration or manual dismissal, and None, Sum, or Replace merging. Let the emitter own output space, target following, screen projection, camera facing, stacking, prewarming, capacity, and overflow.

Design the event, not just the number. A critical hit can use a sharper profile and stronger color, healing can rise gently toward the player, status text can remain manually dismissible, and combo feedback can merge within a deliberate window instead of creating an unreadable cloud.

```
TextStudioEmissionHandle hit = emitter.EmitValue(damagePreset, amount, target);
if (hit.IsValid && damageChanged) hit.SetValue(updatedAmount);
// Dismiss manually only when this presentation is not time-limited.
hit.Dismiss();
```

LONG TEXT, CODEX PAGES, AND CREDITS

Page Navigator and Auto Scroll Follower keep long content inspectable. Reveal only the visible page, preserve history, and test changing layout at narrow widths.

LOCALIZATION, RTL, AND EXTERNAL DIALOGUE

Keep effect tags around semantic phrases rather than fixed glyph positions. Text Studio preserves resolved text, supports RTL-aware flow, and can receive content from dialogue and localization integrations.

UI TOOLKIT

Animated UI Toolkit labels now share the same profile-owned effects, typewriter motion, timing, start, loop, and skip behavior as TextMeshPro text. They play automatically when attached; deterministic manual stepping remains available for custom clocks and tests.

UI Builder can author markup, profile, playback, reveal, speed, and loop directly in UXML. Ready-to-copy patterns cover complete UXML, code-created labels, ordinary Label bindings, hover restarts, and focus reveals. Frequently changing labels coalesce replacements until the next update and bypass markup work when no Text Studio features are present.

COMBAT FEEDBACK RECIPE

1. Create one Text Emitter and choose a Canvas, world-space, or complete 3D host for the visual result you want.
2. Assign a semantic Emission Preset such as Damage or Critical, then pair it with a readable profile and short Reveal treatment such as Elastic or Pop.
3. Configure the numeric template and shared number format, then keep multi-digit values grouped so each event reads as one unit.
4. Call EmitValue(amount, target). Use Sum or Replace only when the preset, target, group, and merge window describe one deliberate gameplay event.
5. Test stacking, projection, aggregation, overflow, manual dismissal, simultaneous hits, Reduced Motion, and the smallest target font size. The emitter returns the host after its normal Hide lifecycle completes.

LIVE HUD RECIPE

1. Use SetTextDeferred when the same label may change several times before the next visual update.
2. Keep the default state still and use a brief effect only when the value meaningfully changes.
3. Use source-color or authored ranges to isolate units, warning thresholds, or status words.
4. Avoid rebuilding markup or changing shared profiles every frame.

SUBTITLE AND ACCESSIBILITY RECIPE

1. Start with By Word or a readable glyph timing preset.
2. Use restrained reveal motion, a stable background, sufficient contrast, and a No Motion path.
3. Keep authored tags around semantic phrases rather than fixed glyph positions so localization remains valid.
4. Make waits skippable, expose a continue action, and test long translations at narrow widths.
5. Verify that speaker names, portraits, and dialogue progression remain owned by the dialogue system.

KARAOKE RECIPE

1. Drive a Color Sweep or Shimmer layer through external progress.
2. Use source-color selection when speaker colors or lyric sections must remain distinct.
3. Apply progress to the intended range and resubmit the external effect while it is active.
4. Add a short release tail only if it improves readability after the sung position passes.

PRODUCTION DIALOGUE RECIPE

1. Let the dialogue system select the next line and Text Studio present the current line.
2. Route content through TextStudioDialogueSource and keep direct TMP writes disabled.

3. Use markers for camera, portrait, audio, or quest beats, not for story branching ownership.
4. Bind continue input to AdvanceOrSkip so the first press completes the line and the next press advances.
5. Test interruption, disable or enable, repeated lines, locale changes, waits, skip, and end-of-conversation cleanup.

LEARN THROUGH THE SAMPLES

The samples form a path rather than a feature dump. Begin with the numbered Learn scenes, move into production dialogue, subtitles, HUD, and long-text examples, then study the Whisperwood capstone scenes to see several systems working together.

- Learn scenes: one concept at a time, including markup, areas, typewriter, actions, placement, corner deformation, and reusable curves and playback assets.
- Production scenes: complete dialogue and subtitle patterns, pooled world-space combat feedback, projected feedback for comparison, timed combo tiers, HUD notifications, and interaction patterns.
- Whisperwood: a holistic demonstration of pacing, effects, input, markers, sound, and presentation.
  PlayMaker 2: an inspectable Whisperwood FSM owns dialogue flow, input, variables, waits, callbacks, and event routing while Text Studio owns the rendered line.
  PrimeTween: a UI Toolkit notification combines Text Studio glyph animation with whole-element movement, rotation, scale, and opacity.
- Text Gallery: searchable starting objects and editable examples.
- Recipes: focused setups for dialogue, long text, audio cues, markers, and integrations.

CONNECT OTHER TOOLS

Optional integrations are imported only when both tools are installed. The base package remains complete without them.

- Timeline for authored sequences.
- Input System for wait-for-input actions.
- Adventure Creator, Game Creator 2, Ink, Naninovel, Pixel Crushers, and Yarn Spinner for dialogue flow.
- Unity Localization for localized content.
  UniText 3 for advanced Unicode shaping, bidirectional layout, fallback fonts, color glyphs, and animation without a TextMeshPro component.
  UIEffect for TextMeshPro transitions that remain visible and follow animated text in Live Preview.
- Unity Visual Scripting and Game Creator 2 for events and triggers.
- LED Studio as an optional renderer.
  PlayMaker 2 for typed FSM actions, callback waits, markers, and optional always-on event routing.
  PrimeTween for whole-element UI Toolkit motion around Text Studio animated labels.

INTEGRATION INSTALLATION MODEL

Text Studio integrations are optional nested packages. Install Text Studio and the companion first, then import the matching integration package. Guarded assemblies compile only when their required package version or verified source symbol is present. Removing an integration must leave base Text Studio free of missing scripts and missing types.

One system should own content progression. Text Studio owns markup, reveal, effects, placement, presentation, and renderer output for the current line. The companion continues to own its narrative state, localization tables, timeline, UI lifecycle, board scheduling, or other domain behavior.

Integration	Requires	Companion owns	Text Studio owns
Input System	Input System 1.4.0+	Input actions and device bindings	Keyed wait and skip lifecycle
Timeline	Timeline 1.7.0+	Clip sequencing and playable time	External effect evaluation
Unity Splines	Splines 2.0.0+	Spline assets and knots	Text selection, effects, and placement dispatch
Unity Localization	Localization 1.0.0+	Tables, entries, and locale	Localized markup presentation
Visual Scripting	Visual Scripting 1.0.0+	Graph flow	Text commands, state, and events
Adventure Creator	Verified 1.86.2 source	Speech, menus, and narrative flow	Subtitle reveal and effects
Game Creator 2	Verified source package	Instruction and trigger flow	Text commands and callbacks
PlayMaker 2	2.0.0-beta.77 to <3.0.0	FSM state, input, variables, and event routing	Markup, reveal, effects, waits, and callbacks
Ink	Ink Unity Integration	Story state and choices	Line reveal and markup
Naninovel	Naninovel 1.21.0+	Script and printer flow	Revealable text presentation
Pixel Crushers	Dialogue System 2.2.70	Conversation and panel lifecycle	Subtitle reveal and effects
Yarn Spinner	Yarn Spinner 3.2.4+	Dialogue execution and options	Standard presenter, reveal, and markup

Integration	Requires	Companion owns	Text Studio owns
LED Studio	LED Studio 1.0.0+	Board scheduling and scrolling	Animated text capture
PrimeTween	com.kyrylokuzyk.primetween	Whole-element motion and opacity	Glyph markup, reveal, motion, and color
UIEffect	UIEffect 5.0.0+	UI proxy effects and transitions	Animated TMP mesh source
UniText 3	3.0.0-pre.6 to <4.0.0	Shaping, order, and fallback fonts	Semantic animation and presentation

Open Tools > Text Studio > Integrations to search the complete support catalog. Each card reports package, define, sample, selected-object, and setup readiness, then offers the relevant setup, fix, sample, or documentation action. Configure Selected is the fastest path when an integration needs components or serialized wiring on the current object.

> The Integrations hub and Health Check use the same catalog. If one reports that a package or symbol is missing, resolve that readiness issue before debugging the runtime scene.

INPUT SYSTEM

Requires Input System 1.4.0 or newer. The guarded TextStudio.InputSystem assembly supplies keyed input through InputActionReference mappings. Map project names such as submit or next, optionally auto-enable the actions, and use the same key in <waitForInput input=submit timeout=5>. The wait resumes on the mapped action, timeout, or cooperative skip. Missing mappings should fall back deliberately or produce a setup warning.

TIMELINE

Requires Timeline 1.7.0 or newer. Blendable reveal and effect clips drive Text Studio from Timeline time while ordinary markup effects remain active underneath. Add Text Studio Control markers for one-shot text changes, reveal, waits, events, disappearance, and effect commands without creating a clip for each cue. Overlaps respect effective weights and remain stable while scrubbing, pausing, seeking, or playing in reverse. Use Timeline for authored cinematic timing and the component API for gameplay-driven effects.

UNITY SPLINES

Requires Unity Splines 2.0.0 or newer. SplinePlacementProvider contributes a package-owned placement choice while Text Studio retains source, effects, timing, selection, serialization, and dispatch. The shipped showcase persists three editable spline containers and provider setups. Use this integration to arrange glyph origins along a scene-authored curve.

UNITY LOCALIZATION

Requires Unity Localization 1.0.0 or newer. TextStudioLocalizeString resolves a LocalizedString into Text Studio content and can restart reveal after a locale change. TextStudioLocaleSwitcher provides sample locale selection. Localized entries may contain Text Studio markup, but each locale must preserve valid tag structure and semantic ranges. Validate missing tables, missing entries, fallback fonts, RTL layout, and repeated enable or disable.

UNITY VISUAL SCRIPTING

Requires Unity Visual Scripting 1.0.0 or newer. The integration exposes curated units for show, set, append, clear, reveal, hide, waits, speed, progress, loop, state, and markers. TextStudioVisualScriptingEvents forwards shown, disappeared, marker, wait-started, and wait-finished events. Enable glyph events only when a graph truly needs their high-frequency callbacks. Wait units require a coroutine-capable graph root.

ADVENTURE CREATOR

The verified source integration uses Adventure Creator 1.86.2 with the TEXTSTUDIO_ADVENTURE_CREATOR symbol. Add TextStudioAdventureCreatorSpeechBridge to the subtitle Canvas and route the visible line through TextStudioDialogueSource. Adventure Creator owns speech, menus, audio, translations, action lists, and final line advance. Text Studio owns the complete subtitle string, markup, reveal, effects, markers, glyph audio, and TMP output.

Do not let an AC MenuLabel write the same visible TMP field. Use TextStudioOwnsReveal for new projects. AdventureCreatorScrollSync exists for an established project that must mirror AC character progress through DriveRevealProgress. Run the setup validator and test speech start, complete, skip, repeated lines, and menu teardown.

GAME CREATOR 2

The Game Creator 2 integration supplies complete visual-scripting coverage for text, reveal, disappearance, effects, waits, events, speed, progress, callbacks, and state conditions. TextStudioGameCreator2EventRelay routes completion or markers into a local Trigger, while the optional Dialogue skin adapter keeps Game Creator actors, portraits, choices, and story timing intact and lets Text Studio remain the only visible subtitle writer.

PLAYMAKER 2

Requires PlayMaker 2 2.0.0-beta.77 or newer and below 3.0.0. Typed actions under Text Studio/PlayMaker 2 control content, reveal, hide, waits, effects, speed, state, and dialogue selection. State listener actions wait for Text Studio callbacks while their FSM state is active. Add

TextStudioPlayMaker2EventRelay only when show, disappear, marker, wait, or optional per-glyph events must always reach one chosen FSM.

Open the Whisperwood demo and inspect Dialogue Director. Its parallel Dialogue Flow and Controls and Event Monitoring regions keep branching, input, variables, and event routing in PlayMaker while Text Studio owns markup, reveal, effects, waits, markers, and disappearance.

INK

Install Ink Unity Integration, import the integration, add TextStudioInkStoryBridge next to TextStudioDialogueSource, assign compiled Ink JSON, and call StartStory. Ink owns story state, continuation, choices, and tags as narrative metadata. Text Studio owns the visible line, markup, reveal, effects, and continue-or-skip presentation. SaveStateJson and LoadStateJson let save slots, checkpoints, and scene transitions preserve the exact Ink story position without serializing Text Studio presentation state.

Bind choice buttons to ChooseChoiceAt(index). AdvanceOrSkip completes the current reveal but refuses to advance while choices are pending. Avoid raw Text Studio hexadecimal color tags in .ink source because Ink treats # as its tag syntax; use named colors or a verified escaping workflow.

NANINOVEL

Requires Naninovel 1.21.0 or newer. TextStudioNaninovelRevealableText implements the revealable-text target used by a Naninovel printer. The markup compiler and preprocessor translate compatible Naninovel markup and preserve source ranges. Naninovel owns script execution, printer visibility, authors, avatars, input indicators, and continue flow. Text Studio owns visual reveal, effects, and rendered text.

PIXEL CRUSHERS DIALOGUE SYSTEM

The verified source workspace uses Dialogue System 2.2.70 and the TEXTSTUDIO_PIXEL_CRUSHERS symbol. Replace StandardUISubtitlePanel with TextStudioPixelCrushersSubtitlePanel and assign or colocate TextStudioDialogueSource. Pixel Crushers owns conversation state, panel lifecycle, portraits, response menus, accumulation, scrolling, sequencing, and final advance. Text Studio owns subtitle markup, reveal, effects, sounds, and skip.

Remove competing vendor typewriter behavior from the routed text. The bridge keeps the vendor subtitle mirror updated for accumulation and scroll checks while preventing duplicate reveal playback.

YARN SPINNER

Requires Yarn Spinner 3.2.4 or newer. Select Yarn's standard LinePresenter, open the Integrations hub, and use Configure Selected. Text Studio adds its dialogue source, custom TextStudioYarnTypewriter, and TextStudioYarnMarkupProcessor without replacing Yarn's presenter. Yarn keeps character names, fades, story flow, commands, options, localization, and advancement. Text Studio owns reveal, supported replacement markup, effect tags, timing, actions, markers, and TMP output. Hurry-up completes the current reveal; next-content dismisses the standard presenter so Yarn can continue.

LED STUDIO

Requires Text Studio 1.0.1 and LED Studio 1.0.0 or newer. LedTextStudioRenderBackend is both a Text Studio renderer and external source bridge for LED Studio. LED Studio owns source scheduling, scrolling, seamless wrap, blinking, and board updates. Text Studio owns markup processing, reveal, effects, placement, presentation, and formatting. The bridge captures the final animated TMP surface into the LED board without reimplementing glyph animation math.

PRIMETWEEN

Install PrimeTween from com.kyrylokuzyk.primetween, then import the Text Studio PrimeTween integration. The shipped UI Toolkit sample attaches Text Studio to ordinary Label elements while PrimeTween moves, rotates, scales, and fades the containing card. Keep glyph markup, reveal, motion, and color in Text Studio; keep whole-element transforms and opacity in PrimeTween.

Unity 2022.3 through 6000.1 uses UI Toolkit Basic mode, where markup, typewriter reveal, actions, events, and PrimeTween element animation remain available. Unity 6000.2 and newer uses Full Parity mode and also applies Text Studio per-glyph motion and color to native text vertices.

UIEFFECT

Requires UIEffect 5.0.0 or newer and TextMeshProUGUI. Text Studio writes the animated TMP vertices, then the integration marks UIEffect's proxy mesh dirty after upload. The next Canvas rebuild consumes the current animated mesh, including UIEffect transition coordinates and expanded geometry, without creating a TMP regeneration loop. Validate both Raw and Live Preview with an active UIEffect transition.

UNITEXT 3

Requires UniText 3.0.0-pre.6 up to, but not including, 4.0.0. TextStudioUniTextBackend lets UniText own shaping, fallback fonts, bidirectional visual order, ligatures, decorations, and

generated glyph geometry while Text Studio owns semantic markup, reveal, effects, placement, and presentation.

The adapter publishes shaped glyph IDs, clusters, advances, font data, visual order, source colors, decorations, and metrics through renderer-neutral contracts. Use it for Arabic, Hebrew, mixed scripts, ligatures, fallback fonts, and other content where simple character-to-glyph assumptions are insufficient. Text Studio 3D can consume the same shaped-glyph contract when its separate sample package is installed.

INTEGRATION CAPABILITY CHECKLIST

- Confirm both products and the integration package are installed in the documented order.
- Confirm the guarded integration assembly is present and the base package still compiles when it is absent.
- Assign exactly one source owner and one visible renderer owner.
- Exercise show, reveal, skip, wait, completion, interruption, disable or enable, and repeated-use paths.
- Open the shipped sample or setup scene and compare its serialized wiring with the production scene.
- Record the installed companion version and include it with support requests.

DIAGNOSE PROBLEMS

START WITH HEALTH CHECK

Open Tools > Text Studio > Health Check. Resolve missing TMP setup, duplicate event systems, unavailable renderers, invalid effect tags, and profile ownership before tuning visuals.

IF THE SOURCE AND PREVIEW DISAGREE

- Confirm whether Raw or Live is selected.
- Check profile ownership and local override mode.
- Verify markup delimiters and closing tags.
- Use Debug to inspect resolved text, visible glyphs, effect bindings, and waits.
- Check renderer compatibility when a composed effect was authored for another space.

IF AN EFFECT IS TOO STRONG

Reduce amplitude, rotation, travel, flash frequency, or spatial radius before changing the content. Test Full, Reduced, and No Motion. Judge the smallest text size and busiest background used by the game.

TROUBLESHOOTING BY SYMPTOM

TAGS APPEAR AS TEXT

Check the authoring profile, delimiters, spelling, and whether the tag belongs to the current catalog. Unknown Text Animator braces intentionally remain literal. Use the tag browser and Debug parsed-occurrence list to distinguish an unknown tag from a renderer passthrough tag.

REVEAL DOES NOT START

Check the Typewriter settings source, reveal mode, start behavior, current pause depth, active action waits, external progress, and whether the line was set through Text Studio. GetRevealDebugSnapshot reports progress, glyph count, active actions, and current state.

CONTINUE SKIPS TWO LINES

Bind the button to AdvanceOrSkip and request the next line only when it returns true. Do not call SkipReveal and advance the narrative system from the same first press.

THE EFFECT WORKS IN ONE RENDERER ONLY

Inspect the effect's declared render domain and channels. Transform, color, and alpha are broadly portable. Material, UV, surface-mask, topology, or 3D channels need a renderer that explicitly supports them.

LOCALIZED OR RTL TEXT BREAKS A RANGE

Keep tags around semantic phrases in every localized entry. Validate final shaping and fallback fonts with the actual backend. Do not derive effect ranges from fixed source character offsets when the language or shaping process can change glyph count or visual order.

PERFORMANCE CHANGES AFTER LIVE UPDATES

Use SetTextDeferred for values replaced several times before a frame. Keep shared profiles and catalogs stable. Avoid repeated registration, source scanning, or allocation-heavy glyph callbacks. Measure a representative line with the real renderer, effects, placement, language, and update frequency.

MOVE AN EXISTING PROJECT TO TEXT STUDIO

Open Tools > Text Studio > Setup > Migration when Text Animator or TMPEffects markup is present. Choose the source provider, load a selected Text Studio component or TextAsset, compare provider markup with canonical Text Studio output, and review source-ranged diagnostics before copying or applying with Undo. Use Advanced Text Animator Project

Migration only when components, styles, timing assets, events, or audio bindings also need conversion.

- Keep a source-control checkpoint before converting.
- Resolve unsupported tags and aliases in the report instead of guessing.
- Compare reveal timing, visible text, area grouping, and skip behavior.
- Keep old and new components out of the same final text object.
- Run Health Check after the last migrated scene.

REFERENCE AND DEVELOPER APPENDIX

PERFORMANCE AND FREQUENT UPDATES

Frequently changing TextMeshPro counters, labels, documents, chat text, and UI Toolkit labels can coalesce replacements until the next update. Request an immediate update only when same-call visibility is required. For temporary text, prewarm the emitter during a loading or setup moment and size inactive retention separately from Maximum Active. The emitter uses one Unity object pool per complete host prefab, keeps returned hosts hidden so renderer state can stay warm, and releases every retained instance when the emitter is disabled or destroyed. Feature-free persistent text still uses a direct backend path, animated TextMeshPro can reuse the freshly generated source mesh, compatible whole-text and multi-effect stacks avoid duplicate glyph work, and stepped effects refresh only when their visible state changes.

PREVIEW EXPORT

Preview export can produce 960 by 540 MP4 at 60 frames per second on Windows and macOS, or transparent GIF at 50 frames per second. Linux supports GIF but not MP4. Export can use flat text or the installed renderer, including lit perspective 3D. Frame planning distinguishes the last visible glyph, reveal completion after its trailing wait, and the point when appearance effects settle, with Reveal Speed scaling applied to each milestone.

RUNTIME CONTROL

For scripted experiences, keep the public surface small: set text, start or skip reveal, play hide, choose a profile, and listen for waits, markers, and completion. IsWaitingForAction reports whether a timed or input action currently owns the wait, and SkipCurrentWait() cancels that wait without confusing it with an ordinary pause. Detailed class and extension contracts are documented in the package API comments and custom-effects guide.

Temporary text uses the same small surface. EmitValue and EmitText return a generation-safe TextStudioEmissionHandle. A live handle can update text or value and request dismissal; after release or pool reuse, the same calls safely return false.

```
TextStudioEmissionHandle score = emitter.EmitValue(scorePreset, 1250, target);
score.SetValue(1500);
score.SetText("NEW HIGH SCORE");
score.Dismiss();
```

CUSTOM EFFECTS AND CHANNELS

Create a custom effect only when built-in tags and Composed Effects cannot express the result. Declare its channels, phases, supported renderer spaces, motion behavior, determinism, and flash characteristics so the editor can give accurate compatibility guidance.

SUPPORT

When requesting support, include the Unity version, Text Studio version, render pipeline, a minimal sample, the Health Check result, and the exact source text that reproduces the issue.

ESSENTIAL SCRIPTING API

TextStudioText is the authoritative component facade. TextStudioApi is the task-oriented convenience layer for common setup and playback. Both accept normal Unity lifecycle rules and must be called from the main thread when they touch components or Unity objects. TextStudioApi treats a missing hub as a safe no-op.

```
using TextStudio;

public void ShowReward(TextStudioText label, string itemName)
{
    TextStudioApi.ConfigureReveal(label, RevealMode.ByGlyph, 1.25f);
    TextStudioApi.ShowText(label,
        $"You found a <shimmer>{itemName}</shimmer>!");
}
```

SetText changes the authored source. StartReveal restarts from the beginning. ContinueReveal continues current progress. SkipReveal completes the line. PauseReveal and ResumeReveal use a balanced pause depth. SkipCurrentWait requests cooperative cancellation for a timed, input, or audio wait. AdvanceOrSkip returns true only when the caller may request the next line.

```
if (text.IsWaitingForAction)
    text.SkipCurrentWait();
else if (text.IsRevealing)
    text.SkipReveal();
else
    RequestNextLine();
```

SetRevealSpeed clamps the multiplier to at least 0.01. SetRevealProgress and DriveRevealProgress accept normalized progress. SetGlyphVisibility, SetWordVisibility, and SetTextVisibility override visibility without changing typewriter progress and report invalid requests with false. ClearVisibilityOverrides returns ownership to reveal.

EXTERNAL EFFECTS

ApplyExternalEffect submits one renderer-neutral layer for the current evaluated frame. Submit it every frame while active. Progress and intensity are clamped. Omitted glyph bounds address all

visible glyphs, explicit bounds are inclusive, invalid requests return false, and accepted layers run in submission order after ordinary effects.

```
void Update()
{
    float progress = Mathf.Repeat(Time.time, 1f);
    text.ApplyExternalEffect(effect, progress, 1f);
}
```

UI TOOLKIT ESSENTIALS

Use TextStudioAnimatedLabel in UXML or TextStudioUITKBinding for an existing TextElement. Automatic mode owns one scheduled item, advances at most once per Unity frame, sleeps when no time-dependent feature is active, and stops on panel detach. Manual mode is for deterministic host clocks, tests, or custom scheduling.

```
TextStudioUITKBinding binding = label.AttachTextStudio(profile);
binding.SetTextDeferred("Score <pulse>1250</pulse>");
// In Manual mode only:
binding.UpdateFromHost(deltaTime);
```

REGISTRATION AND EXTENSION ESSENTIALS

Register custom effects, actions, curves, playbacks, timings, expansion sheets, extension tags, renderers, text sources, and placement providers only through their documented interfaces. Use stable namespaced IDs. Parameter parsing belongs at bind time, not inside per-glyph Evaluate calls. Transient plan and frame views must not be retained beyond their callback.

```
TextStudioApi.RegisterEffect(new OrbitEffect());
TextStudioApi.RegisterAction(customAction);
TextStudioApi.RegisterCurve("studio.ease", curve);
```

Full supported API browser: https://www.wetzold.com/tools/text-studio/api/. The browser should be used for complete signatures, parameters, return values, lifecycle notes, and extension contracts.

INSPECTOR AND ASSET REFERENCE

- TextStudioText inspector: Content, Typewriter, Placement, Events, Debug, and Preview.
- TextStudioProfile: default effects, typewriter recipe, source ownership policy, and reusable object defaults.
- TextStudioTypewriterPreset and Motion Preset: reusable pacing, lifecycle, motion, skip, wait, and hide behavior.
- CharacterTimings and WordTimings: delay and punctuation rules.
- TextStudioComposedEffectAsset: ordered layers, phases, curves, playback, scope, influence, sharing, and preview.
- Effect Curve and Effect Playback: reusable progress and time envelopes.
- TextStudioMarkupSyntax and Tag Expansion Sheet: authoring grammar and project vocabulary.
- TextStudioAudioSet, Audio Player, Play Sound assets and components: glyph, marker, and authored sound behavior.
- Marker Event Router, Typewriter Event Relay, Dialogue Source, Page Navigator, and Auto Scroll Follower: production flow helpers.
- Project Settings, Health Check, Migration, Gallery, Recipes, Tag Browser, and Effect Code: project-wide authoring and diagnostics.

Inspector coverage, part 1 of 5

Inspector or surface	Controls and ownership	Primary manual coverage
TextStudioText main inspector	Persistent preview plus Content, Typewriter, Placement, Events, and Debug tabs.	First result; runtime model; reveal; placement; actions; troubleshooting
Persistent preview controls	Raw or Live, phase selection, speed, repeat, scrub, skip, hide, waits, and renderer preview.	Authoring modes; runtime model; preview and export
Content tab	Source, profile, default effects, markup help, backend state, and diagnostics.	Markup; effects; profiles; backends
Typewriter tab	Reveal source, unit, timings, clocks, motion, start, loop, skip, waits, and hide.	Build Reveals and Typewriter Experiences
Placement tab	Built-in placement modes, orientation, extension providers, and Spatial Effect Target.	Place and Influence Text
Events tab	Reveal lifecycle, markers, relays, payloads, and dialogue handoff.	Sound, Input, Events, and Interaction
Debug tab	Resolved source, parsed ranges, active bindings, renderer channels, waits, and ownership.	Diagnose Problems
TextStudioDefaultSettings	Project-wide built-in defaults and fallback behavior.	Project and Shared Settings
CharacterTimings	Character categories, punctuation delay, whitespace behavior, and overrides.	Character and Word Timings
WordTimings	Word pacing, punctuation-aware pauses, and readable rhythm.	Character and Word Timings

Inspector coverage, part 2 of 5

Inspector or surface	Controls and ownership	Primary manual coverage
TextStudioTypewriterPreset	Complete reusable reveal and hide recipe.	Reveal units and settings sources; Profiles
TextStudioTypewriterMotionPreset	Reusable reveal or hide motion without changing timing.	Reveal and Hide Motion
TextStudioProfile	Default effects, typewriter recipe, source ownership, and restored local overrides.	Reuse Your Work
TextStudioProjectSettings	Catalogs, syntax, expansion sheets, defaults, motion policy, and validation.	Project and Shared Settings
TextStudioMarkupSyntax	Delimiter pair, native angle-tag acceptance, and literal escaping.	Custom Delimiters and Syntax Assets
TextStudioTagExpansionSheet	Short project vocabulary expanded into reviewed markup stacks.	Tag Expansion Sheets
TextStudioEffectAsset	Reusable effect parameters and renderer compatibility.	Effects; Reference and Developer Appendix
TextStudioComposedEffectAsset	Layer order, channels, phases, targeting, blends, influence, sharing, and diagnostics.	Create Composed Effects
EffectCurveAsset	Reusable normalized progress shape and custom curve.	Phases, Curves, Playback, and Phase Shift
EffectPlaybackAsset	Once, loop, ping-pong, persistence, duration, delay, and phase behavior.	Phases, Curves, Playback, and Phase Shift

Inspector coverage, part 3 of 5

Inspector or surface	Controls and ownership	Primary manual coverage
TextStudioAudioSet	Glyph clips, marker cues, source choice, pitch, volume, and cooldown policy.	Audio Sets and Audio Player
TextStudioAudioPlayer	Runtime clip selection, overlap, interruption, marker routing, and ownership.	Audio Sets and Audio Player
Play Sound asset and component	Authored one-shot sound, optional wait for completion, and scene-local ownership.	Built-in action tags; Audio Sets and Audio Player
Custom action asset and component	Reusable or scene-local authored action with cooperative wait and skip behavior.	Actions Inside the Line; Developer Appendix
InputSystemWaitInputAction	Named InputActionReference mapping, enable policy, timeout, and fallback.	Input System integration
TextStudioDialogueSource	Stable content bridge for dialogue, quest, subtitle, or cutscene systems.	Dialogue Source and Long Text
TextStudioMarkerEventRouter	Marker-name routing to UnityEvents or code.	Events, Markers, and Relays
TextStudioTypewriterEventRelay	Reveal and wait lifecycle forwarded without a custom subscriber.	Events, Markers, and Relays
TextStudioPageNavigator	Visible-page navigation without recompiling the complete document.	Long Text, Codex Pages, and Credits
TextStudioAutoScrollFollower	Keeps the newest revealed content visible in a ScrollRect.	Long Text, Codex Pages, and Credits

Inspector coverage, part 4 of 5

Inspector or surface	Controls and ownership	Primary manual coverage
TextStudioSingleEventSystemGuard	Avoids duplicate EventSystems in additive-scene samples.	Samples; Diagnose Problems
TMP setup inspectors	Add Text Studio while preserving source, font, layout, material, and renderer role.	Adopt an Existing TextMeshPro Object
Gallery	Search effects and inspect real rendered starting points.	Learn Through the Samples; Effect Gallery
Recipes	Focused copyable setups for dialogue, HUD, long text, audio, markers, and integrations.	Common Game Scenarios
Examples	Numbered learning path and sample map.	Learn Through the Samples
Welcome	First-run navigation and package verification.	Requirements, Installation, and Verification
Health Check	Project setup, missing resources, renderers, event systems, and integration health.	Start with Health Check
Migration	Reviewed conversion report and staged Text Animator project migration.	Move an Existing Project to Text Studio
Effect Code	Portable treatment payload, dependencies, conflict handling, and import safety.	Effect Code
Preview export	Real renderer, camera, quality, motion, and platform format behavior.	Preview Export

Inspector coverage, part 5 of 5

Inspector or surface	Controls and ownership	Primary manual coverage
UI Toolkit adapters	AnimatedLabel, binding, automatic or manual scheduling, and panel lifecycle.	UI Toolkit
Runtime public API	Text, reveal, waits, visibility, external effects, registration, and transient views.	Essential Scripting API

APPENDIX: EFFECT GALLERY

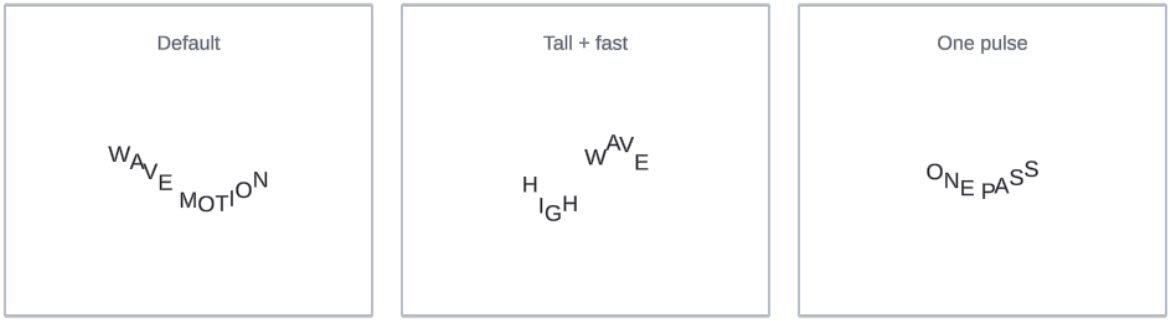
Use this appendix when you want to browse the complete built-in palette. Each card is a real Text Studio render chosen to keep the words readable while still showing the motion or color treatment.

EVERYDAY MOTION

WAVE

Canonical tag: <wave>. Search aliases: none. Key controls and defaults: Amplitude (0.12), Direction (up), Frequency (0.65), Speed (1.05), Duration (1), Playback (infinite).

Travelling motion for magical words and friendly headings. Start with <wave>WAVE MOTION</wave> and tune it in Preview.



Wave: three useful starting points rendered by Text Studio.

SHAKE

Canonical tag: <shake>. Search aliases: jitter, tremble, rumble. Key controls and defaults: Strength (0.08), Speed (10), Progress Index With Time (true), Duration (1), Playback (infinite).

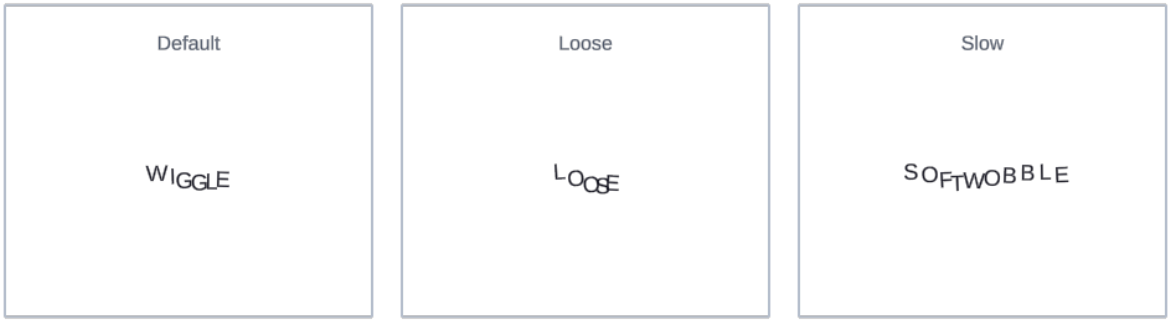
Deterministic impact and instability for damage, warnings, and broken systems. Start with <shake a=.025>UNSTABLE</shake> and tune it in Preview.

Shake: three useful starting points rendered by Text Studio.

WIGGLE

Canonical tag: <wiggle>. Search aliases: wobble, wiggly. Key controls and defaults: Position Strength (0.04), Rotation Degrees (3), Speed (1.2), Progress Index With Time (true), Duration (1), Playback (infinite).

Small organic position and rotation changes for playful text. Start with <wiggle>PLAYFUL</wiggle> and tune it in Preview.

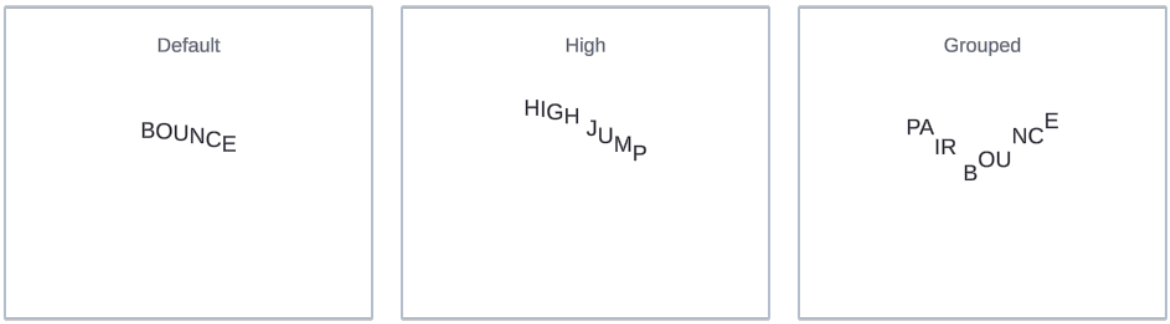


Wiggle: three useful starting points rendered by Text Studio.

BOUNCE

Canonical tag: <bounce>. Search aliases: hop, jump. Key controls and defaults: Height (0.14), Direction (up), Speed (1.05), Stagger (0.045), Duration (1), Playback (infinite).

Lively hops for pickups, rewards, and upbeat prompts. Start with <bounce>BOUNCE</bounce> and tune it in Preview.



Bounce: three useful starting points rendered by Text Studio.

SWING

Canonical tag: <swing>. Search aliases: sway. Key controls and defaults: Max Degrees (12), Speed (1), Duration (1), Playback (infinite), Pivot (new Vector2(0.5, 1)).

Rotational motion for hanging signs and soft menu movement. Start with <swing>SWING</swing> and tune it in Preview.

Swing: three useful starting points rendered by Text Studio.

PENDULUM

Canonical tag: <pendulum>. Search aliases: none. Key controls and defaults: Max Degrees (16), Speed (0.34), Duration (1), Playback (infinite), Pivot (new Vector2(0.5, 1)).

A slower, more uniform hanging motion across the selected range. Start with <pendulum>PENDULUM</pendulum> and tune it in Preview.

Pendulum: three useful starting points rendered by Text Studio.

DANGLE

Canonical tag: <dangle>. Search aliases: hanging. Key controls and defaults: Max Degrees (8), Speed (1), Duration (1), Playback (infinite), Pivot (new Vector2(0.5, 1)), Horizontal (string.Empty), Vertical (string.Empty).

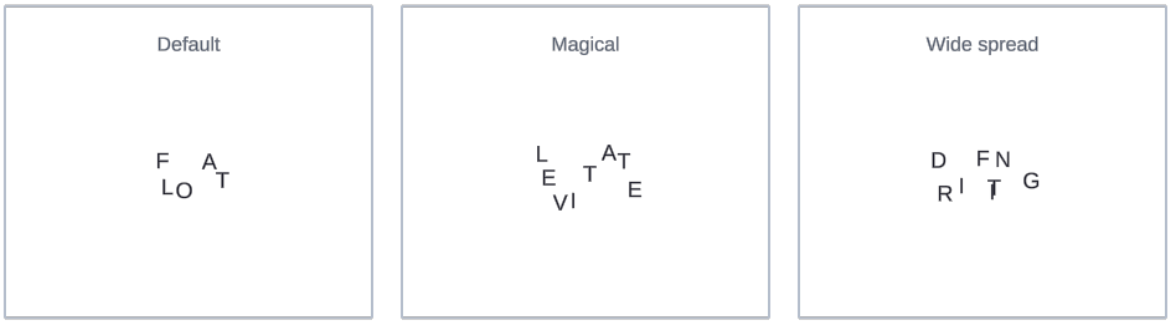
Loose glyphs with staggered hanging movement. Start with <dangle>DANGLE</dangle> and tune it in Preview.

Dangle: three useful starting points rendered by Text Studio.

FLOAT

Canonical tag: <float>. Search aliases: drift, hover, levitate. Key controls and defaults: Vertical Amplitude (0.16), Horizontal Amplitude (0.04), Speed (0.75), Spread (1.25), Duration (1), Playback (infinite).

Ambient drift for magic, dreams, holograms, and underwater text. Start with <float>MAGIC WORDS</float> and tune it in Preview.



Float: three useful starting points rendered by Text Studio.

COLOR AND EMPHASIS

PULSE

Canonical tag: <pulse>. Search aliases: breathe, breathing, beat. Key controls and defaults: Scale Amplitude (0.08), Alpha Amplitude (0.08), Speed (1.35), Glyph Phase Offset (0.18), Duration (1), Playback (infinite).

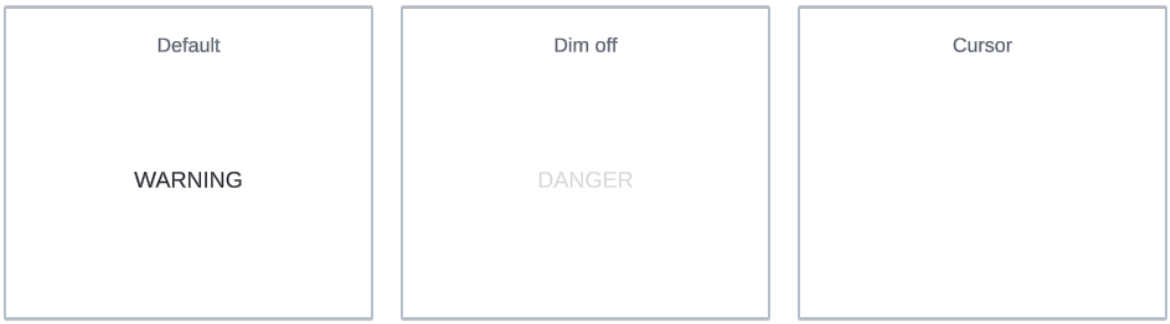
A readable heartbeat for active choices and continue prompts. Start with <pulse>PRESS START</pulse> and tune it in Preview.

Pulse: three useful starting points rendered by Text Studio.

BLINK

Canonical tag: <blink>. Search aliases: flash, flicker. Key controls and defaults: Frequency (1.75), Duty Cycle (0.55), Duration (1), Playback (infinite).

On and off emphasis for cursors and warnings. Use a calm Reduced Motion alternative. Start with <blink>WARNING</blink> and tune it in Preview.

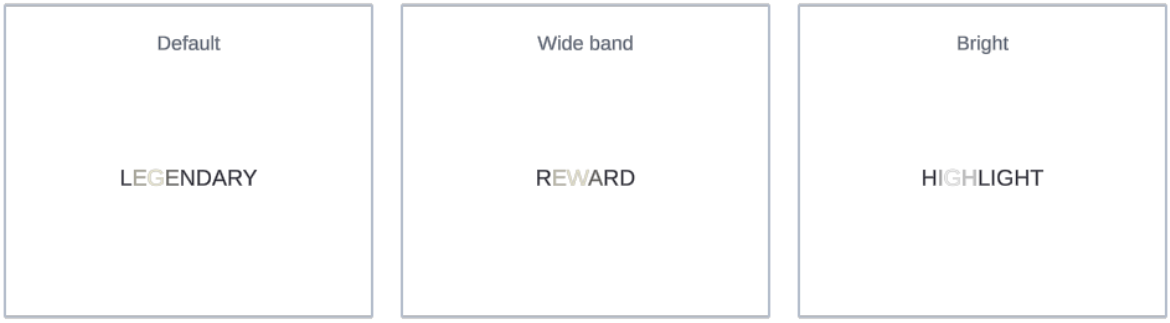


Blink: three useful starting points rendered by Text Studio.

VERTEX SHIMMER

Canonical tag: <shimmer>. Search aliases: shine, sparkle, highlight. Key controls and defaults: Highlight Color (RGBA 255, 246, 190, 255), Speed (0.65), Width (0.16), Scale Boost (0.035), Wrap (true).

A moving highlight band for rare items and premium labels. Start with <shimmer>LEGENDARY</shimmer> and tune it in Preview.



Vertex Shimmer: three useful starting points rendered by Text Studio.

GRADIENT FLOW

Canonical tag: <gradientFlow>. Search aliases: gradient flow, gradient sweep. Key controls and defaults: Color A (RGBA 70, 214, 255, 255), Color B (RGBA 255, 92, 184, 255), Speed (0.35), Spread (1).

An animated two-color treatment across a title or phrase. Start with <gradientFlow>GRADIENT FLOW</gradientFlow> and tune it in Preview.

Gradient Flow: three useful starting points rendered by Text Studio.

Color Sweep

Canonical tag: <colorSweep>. Search aliases: progressive highlight, paint on, karaoke sweep. Key controls and defaults: Color (RGBA 88, 255, 218, 255), Softness (0.14).

A progressive paint-on highlight for karaoke, progress, and guided reading. Start with <colorSweep color=#58ffda>PAINT ON</colorSweep> and tune it in Preview.

Color Sweep: three useful starting points rendered by Text Studio.

Color Pulse

Canonical tag: <colorPulse>. Search aliases: color cycle, palette pulse, tint pulse. Key controls and defaults: Color A (RGBA 255, 255, 255, 255), Color B (RGBA 255, 211, 72, 255), Speed (1.2).

A color-to-color pulse for danger, rarity, magic, and status changes. Start with <colorPulse from=#ff3355 to=#ffffff>CRITICAL</colorPulse> and tune it in Preview.

Color Pulse: three useful starting points rendered by Text Studio.

RAINBOW

Canonical tag: <rainbow>. Search aliases: none. Key controls and defaults: Saturation (0.85), Value (1), Speed (0.25), Spread (0.8).

Celebratory hue movement for arcade and party moments. Start with <rainbow>RAINBOW</rainbow> and tune it in Preview.

Rainbow: three useful starting points rendered by Text Studio.

EMPHASIS

Canonical tag: <emphasis>. Search aliases: em, important, bold. Key controls and defaults: Amplitude (0.08), Speed (4), Duration (1), Playback (infinite).

A gentle pulse that draws attention inside dialogue or instructions. Start with This is <emphasis>important</emphasis> and tune it in Preview.

Emphasis: three useful starting points rendered by Text Studio.

REVEALS AND TRANSFORMATIONS

POP

Canonical tag: <pop>. Search aliases: popIn, punch, overshoot. Key controls and defaults: Start Scale (0.15), Overshoot (0.24).

A punchy overshoot for rewards and short entrances. Start with <pop>BONUS!</pop> and tune it in Preview.

Pop: three useful starting points rendered by Text Studio.

FADE

Canonical tag: <fade>. Search aliases: alpha, fade in, fade out. Key controls and defaults: Max Alpha (1), Duration (1), Playback (pingpong), Color (RGBA 255, 255, 255, 255), Mode (full).

A soft entrance, exit, or breathing treatment. Start with <fade>SOFT REVEAL</fade> and tune it in Preview.

Fade: three useful starting points rendered by Text Studio.

SIZE

Canonical tag: <size>. Search aliases: big, small, grow, shrink. Key controls and defaults: Start Scale (0.85), End Scale (1), Pulse Amplitude (0.08), Pulse Speed (1), Duration (1), Playback (infinite), Scale (string.Empty).

Grow, shrink, or pulse without changing the authored text. Start with <size>BUILD IN</size> and tune it in Preview.

Size: three useful starting points rendered by Text Studio.

ROTATE IN

Canonical tag: <rotate>. Search aliases: spin, rotate in. Key controls and defaults: Start Degrees (45), Speed (0.34), Loop Degrees (360), Axis (z), Duration (1), Playback (infinite), Pivot (new Vector2(0.5, 0.5)).

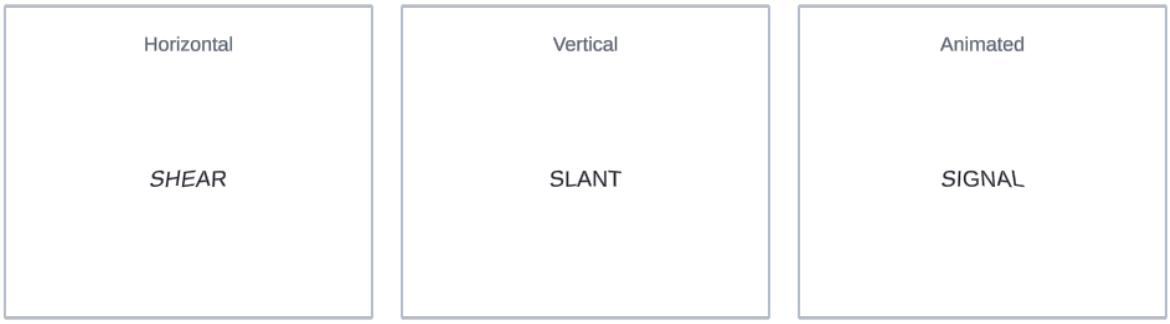
Kinetic rotation for energetic captions and reveals. Start with <rotate>SPIN IN</rotate> and tune it in Preview.

Rotate In: three useful starting points rendered by Text Studio.

SHEAR

Canonical tag: <shear>. Search aliases: skew. Key controls and defaults: Amplitude (1), Horizontal (0), Vertical (allSides), Speed (1), Duration (1), Playback (infinite).

An angled, speed-focused distortion that keeps layout intact. Start with <shear horizontal=5>SHEAR</shear> and tune it in Preview.



Shear: three useful starting points rendered by Text Studio.

SLIDE HORIZONTAL

Canonical tag: <slideHorizontal>. Search aliases: slide, slideH, slideX. Key controls and defaults: Distance (0.22), Speed (1), Stagger (0.03), Duration (1), Playback (infinite), Horizontal (string.Empty), Vertical (string.Empty).

A side entrance or restrained horizontal drift. Start with <slide>FROM LEFT</slide> and tune it in Preview.

Slide Horizontal: three useful starting points rendered by Text Studio.

SLIDE VERTICAL

Canonical tag: <slideVertical>. Search aliases: slideV, slideY, rise, drop. Key controls and defaults: Distance (0.16), Speed (1), Stagger (0.03), Duration (1), Playback (infinite), Horizontal (string.Empty), Vertical (string.Empty).

A rise, drop, or gentle vertical movement. Start with <slideV>RISE UP</slideV> and tune it in Preview.

Slide Vertical: three useful starting points rendered by Text Studio.

EXPAND

Canonical tag: <expand>. Search aliases: scale fade, soft pop. Key controls and defaults: Amplitude (1), Mode (both), Origin (center), Pulse Speed (1), Duration (1), Playback (infinite).

A soft entrance that combines transparency and scale. Start with <expand>EXPAND</expand> and tune it in Preview.

Expand: three useful starting points rendered by Text Studio.

TYPEWRITER

Canonical tag: <typewriter>. Search aliases: typewriter reveal, appear. Key controls and defaults: use the Tag Browser for typed controls.

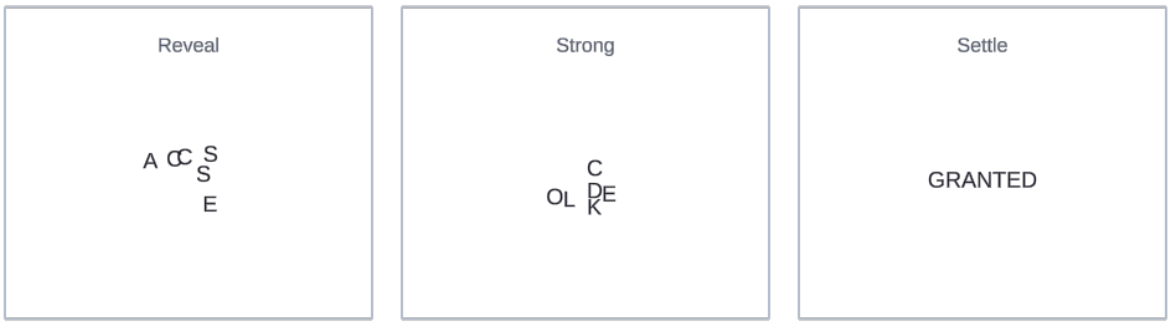
Hides unrevealed glyphs inside the selected range. Start with <typewriter>REVEAL THIS</typewriter> and tune it in Preview.

Typewriter: three useful starting points rendered by Text Studio.

DECRYPT

Canonical tag: <decrypt>. Search aliases: scramble, decode. Key controls and defaults: Shake Strength (0.12), Shake Speed (32).

A terminal-like lock-in during reveal. Start with <decrypt>ACCESS GRANTED</decrypt> and tune it in Preview.



Decrypt: three useful starting points rendered by Text Studio.

GLITCH

Canonical tag: <glitch>. Search aliases: corrupt, digital noise. Key controls and defaults: Strength (0.1), Frequency (8), Seed (2719), Activity (0.18).

Controlled signal loss and digital instability. Start with <glitch>SIGNAL LOST</glitch> and tune it in Preview.

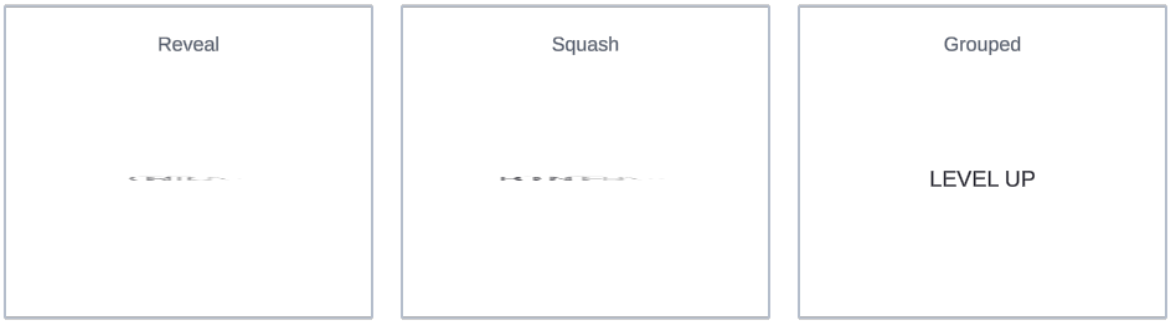
Glitch: three useful starting points rendered by Text Studio.

PRODUCTION 2D

ELASTIC

Canonical tag: <elastic>. Search aliases: squash stretch, jelly. Key controls and defaults: Strength (0.32), Oscillations (2.25), Damping (5.5), Anticipation (0.12), Preserve Area (true).

Squash and stretch for combat text, buttons, rewards, and dialogue beats. Start with <elastic>CRITICAL!</elastic> and tune it in Preview.

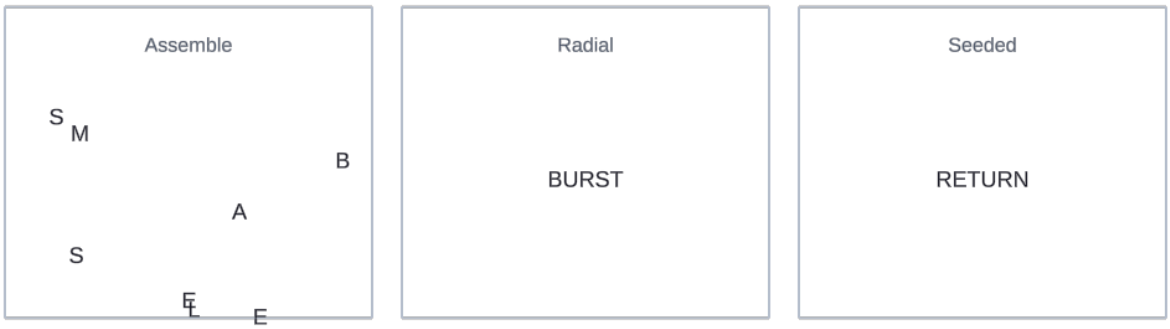


Elastic: three useful starting points rendered by Text Studio.

SCATTER 2D

Canonical tag: <scatter2D>. Search aliases: fly, burst, explode. Key controls and defaults: Distance (0.65), Direction (up), Cone Angle (120), Radial (true), Seed (1729), Overshoot (0.08).

Directional or radial assembly with a repeatable seed. Start with <scatter2D>ASSEMBLE</scatter2D> and tune it in Preview.

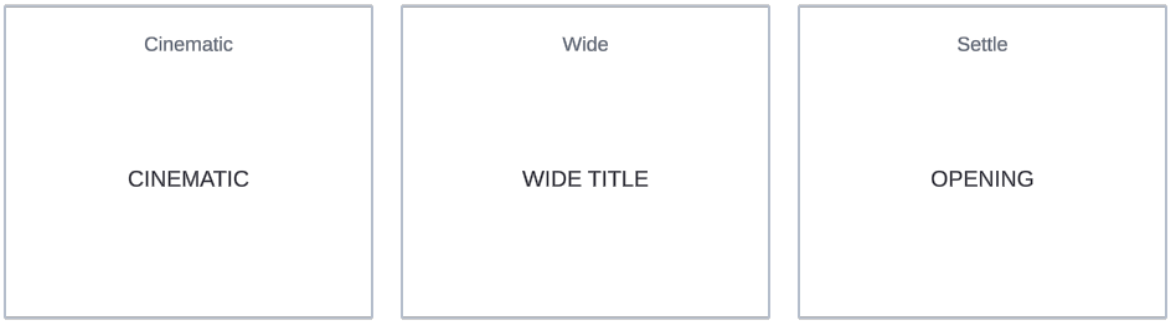


Scatter 2D: three useful starting points rendered by Text Studio.

VISUAL TRACKING

Canonical tag: <tracking>. Search aliases: letter spacing, title tracking. Key controls and defaults: Amount (0.35), Anchor (center).

Cinematic letter-spacing movement without text reflow. Start with <tracking amount=.7>CINEMATIC</tracking> and tune it in Preview.

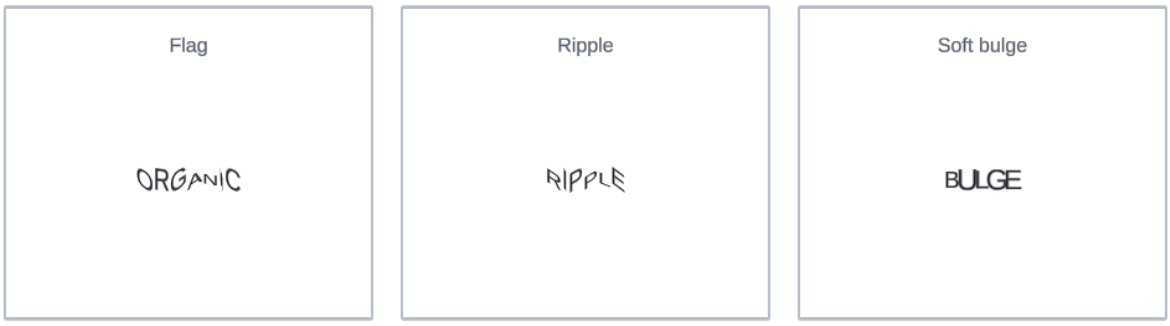


Visual Tracking: three useful starting points rendered by Text Studio.

CORNER WARP

Canonical tag: <cornerWarp>. Search aliases: flag, ripple, quad warp. Key controls and defaults: Mode (ripple), Strength (0.12), Speed (0.8), Spread (1.1).

Flag, ripple, jelly, squeeze, and bulge deformation from one effect. Start with <cornerWarp mode=flag>ORGANIC</cornerWarp> and tune it in Preview.



Corner Warp: three useful starting points rendered by Text Studio.

Focus

Canonical tag: <focus>. Search aliases: none. Key controls and defaults: Amount (1).

A defocused-to-sharp reveal for cinematic titles and target acquisition. Start with <focus>IN FOCUS</focus> and tune it in Preview.

Focus: three useful starting points rendered by Text Studio.

Dissolve

Canonical tag: <dissolve>. Search aliases: none. Key controls and defaults: Softness (0.12), Space (glyph).

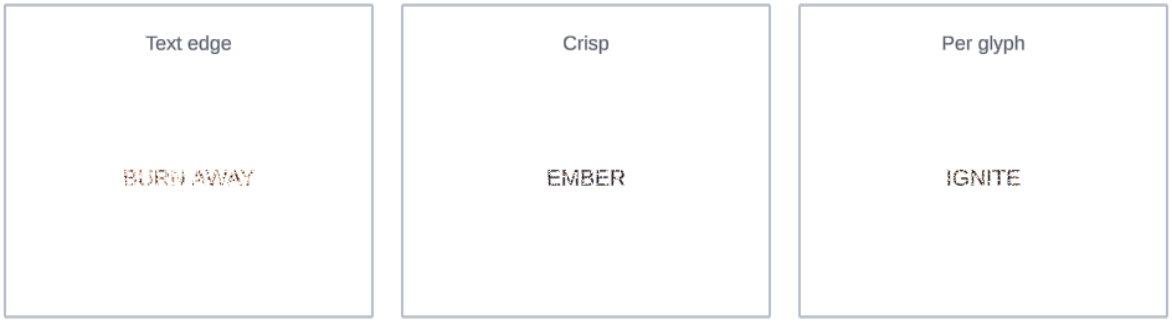
A threshold reveal or hide with an adjustable soft edge. Glyph mapping treats characters independently; Text mapping creates one continuous edge across the complete range. Start with <dissolve softness=.18 space=text>DISSOLVE</dissolve> and tune it in Live Preview.

Dissolve: three useful starting points rendered by Text Studio.

BURN

Canonical tag: <burn>. Search aliases: burn away, embers, char burn. Key controls and defaults: Softness (0.16), Space (text).

A deterministic hot-edge and ember reveal or hide for magical damage, danger, and disintegration. Text mapping travels continuously across the complete range; Glyph mapping burns each character independently. Start with <burn softness=.16 space=text>BURN AWAY</burn> and tune it in Live Preview.

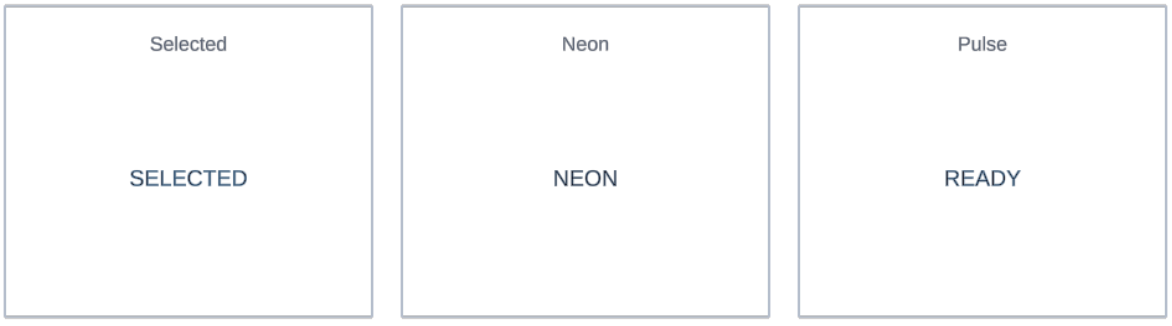


Burn: text-wide, crisp, and per-glyph starting points rendered by Text Studio.

OUTLINE AND GLOW

Canonical tag: <outlineGlow>. Search aliases: none. Key controls and defaults: Outline (0.25), Glow (0.65), Speed (1).

Readable selected, neon, and reward emphasis over busy backgrounds. Start with <outlineGlow>SELECTED</outlineGlow> and tune it in Preview.

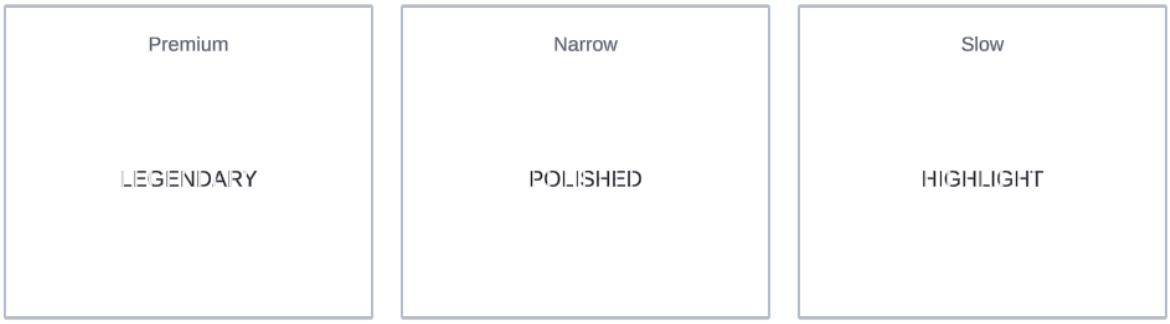


Outline and Glow: three useful starting points rendered by Text Studio.

SURFACE SWEEP

Canonical tag: <surfaceSweep>. Search aliases: none. Key controls and defaults: Speed (0.55), Width (0.15).

A true material highlight for polished titles and premium rewards. Start with <surfaceSweep>LEGENDARY</surfaceSweep> and tune it in Preview.

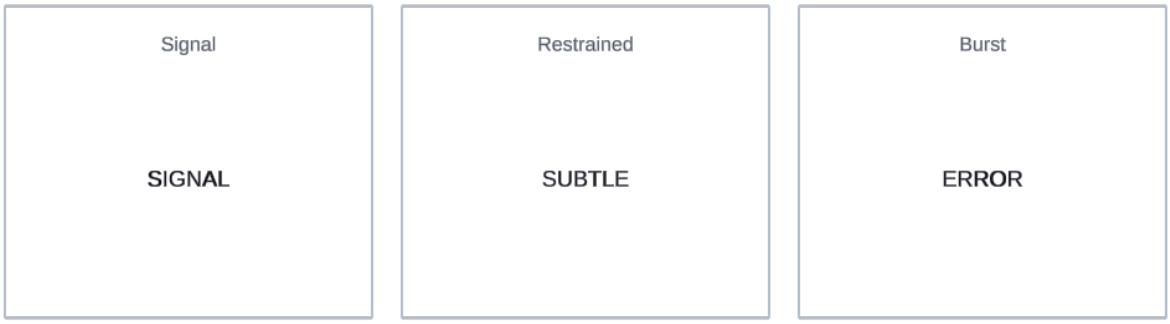


Surface Sweep: three useful starting points rendered by Text Studio.

CHROMATIC GLITCH

Canonical tag: <chromaticGlitch>. Search aliases: none. Key controls and defaults: Strength (0.08), Frequency (7).

Restrained RGB separation and digital slicing. Start with <chromaticGlitch>SIGNAL</chromaticGlitch> and tune it in Preview.



Chromatic Glitch: three useful starting points rendered by Text Studio.

PIXEL GRID

Canonical tag: <pixelGrid>. Search aliases: pixelate, retro, low resolution. Key controls and defaults: Cell Size (8), Active Amount (1), Transition Amount (1), Space (screen), Quality (preserve), Min Cell Size (4), Max Cell Size (10).

True TextMeshPro SDF cell pixelation for retro UI, digital reveals, and stylized game text. Screen space keeps cells aligned to render pixels; Glyph space follows each character; Preserve protects thin strokes, while Crisp favors hard-edged cells. Start with <pixelGrid cellSize=8 space=screen quality=preserve>PIXEL GRID</pixelGrid> and tune it in Preview. Unsupported material and renderer paths keep readable unchanged text.

Pixel Grid: screen-stable, glyph-anchored, and crisp starting points rendered by Text Studio.

SPATIAL MOTION

HERO DEPTH RISE

Canonical tag: <heroDepthRise>. Search aliases: none. Key controls and defaults: Strength (0.75), Lift (0.16), Tilt (24).

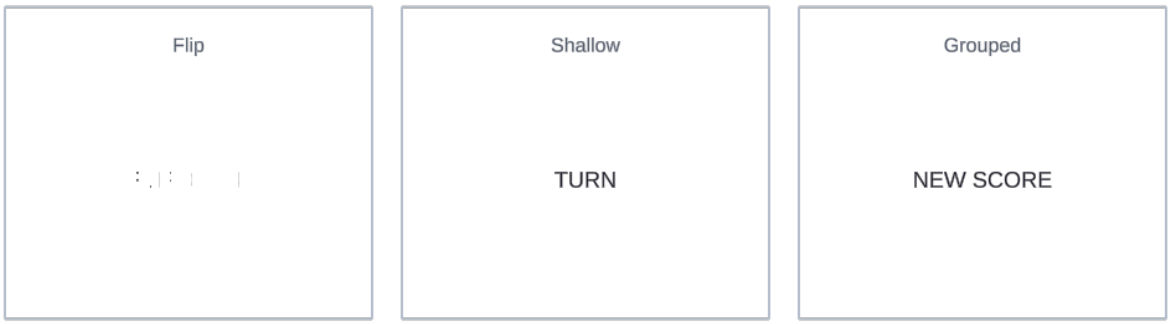
A cinematic rise from depth with a restrained tilt. Start with <heroDepthRise>HERO TITLE</heroDepthRise> and tune it in Preview.

Hero Depth Rise: three useful starting points rendered by Text Studio.

CARD FLIP Y

Canonical tag: <cardFlipY>. Search aliases: none. Key controls and defaults: Angle (92), Depth (0.12).

A vertical-axis turn inspired by premium cards and dimensional tiles. Start with <cardFlipY>FLIP CARD</cardFlipY> and tune it in Preview.



Card Flip Y: three useful starting points rendered by Text Studio.

TOP HINGE

Canonical tag: <topHinge>. Search aliases: none. Key controls and defaults: Angle (-88).

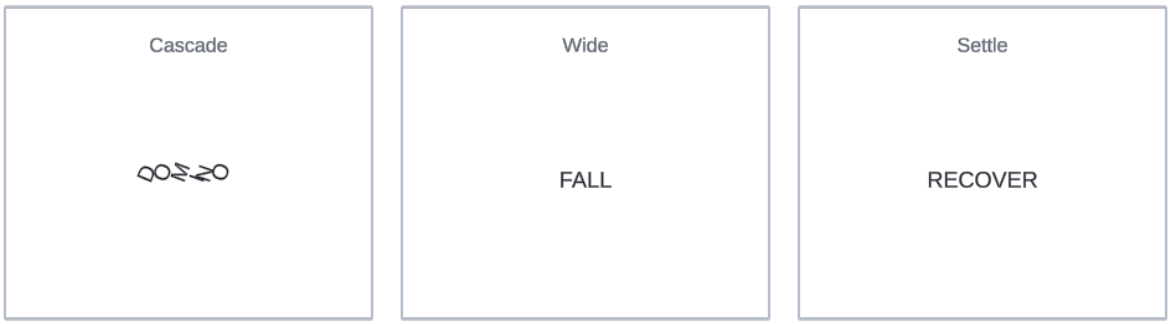
Letters unfold from their top edge. Start with <topHinge>HINGE</topHinge> and tune it in Preview.

Top Hinge: three useful starting points rendered by Text Studio.

DOMINO LINE

Canonical tag: <dominoLine>. Search aliases: none. Key controls and defaults: Angle (72), Depth (0.08).

A stagger-ready sideways fall and recovery. Start with <dominoLine>DOMINO</dominoLine> and tune it in Preview.

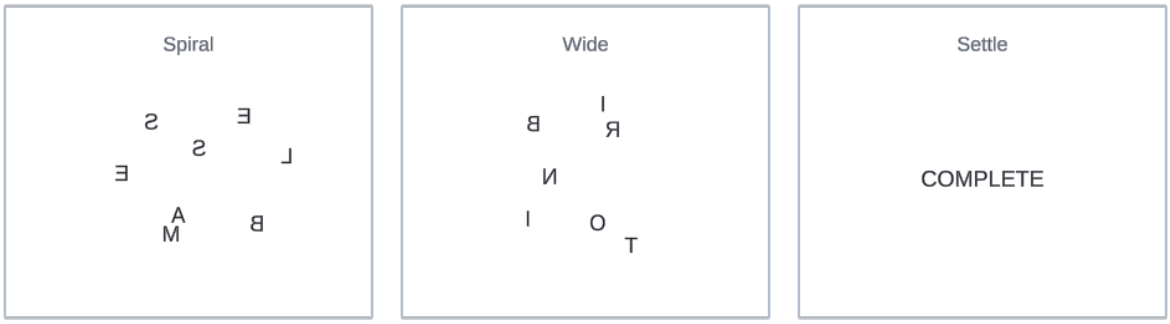


Domino Line: three useful starting points rendered by Text Studio.

SPIRAL ASSEMBLE

Canonical tag: <spiralAssemble>. Search aliases: none. Key controls and defaults: Radius (0.65), Depth (0.8), Turns (1.25).

Letters arrive from a three-dimensional spiral. Start with <spiralAssemble>ASSEMBLE</spiralAssemble> and tune it in Preview.



Spiral Assemble: three useful starting points rendered by Text Studio.

3D SCATTER AND RETURN

Canonical tag: <spatialScatter>. Search aliases: none. Key controls and defaults: Strength (0.85), Seed (1729).

Repeatable scattering through volume followed by a clean return. Start with <spatialScatter>SCATTER</spatialScatter> and tune it in Preview.

3D Scatter and Return: three useful starting points rendered by Text Studio.

Fold Away

Canonical tag: <foldAway>. Search aliases: none. Key controls and defaults: Angle (90), Depth (0.35).

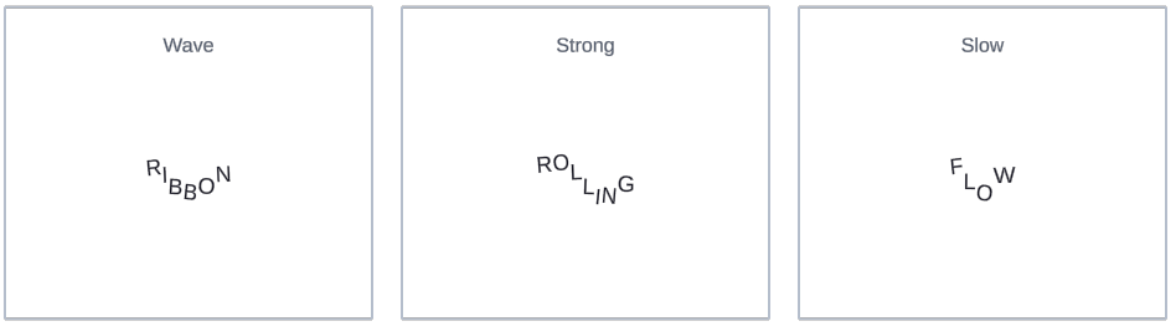
A refined hide that folds letters backward into depth. Start with <foldAway>GOODBYE</foldAway> and tune it in Preview.

Fold Away: three useful starting points rendered by Text Studio.

Rolling Ribbon

Canonical tag: <rollingRibbon>. Search aliases: none. Key controls and defaults: Strength (0.16), Angle (18), Speed (0.75), Spread (0.8).

A broadcast-style wave through position and orientation. Start with <rollingRibbon>RIBBON</rollingRibbon> and tune it in Preview.

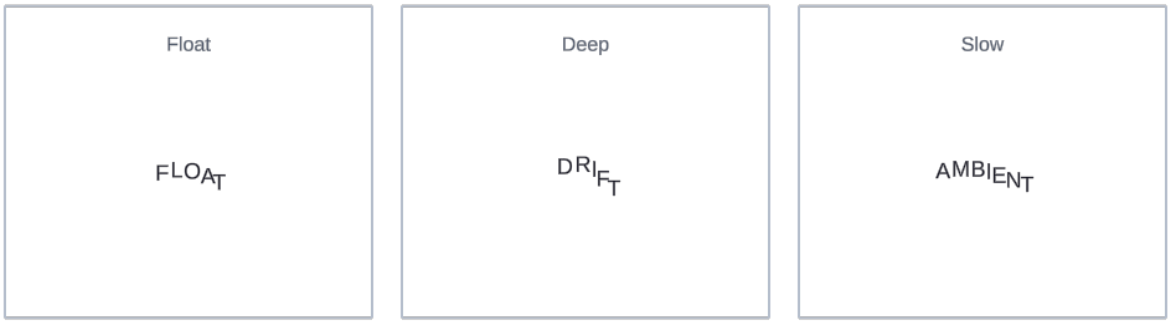


Rolling Ribbon: three useful starting points rendered by Text Studio.

Soft Spatial Float

Canonical tag: <softSpatialFloat>. Search aliases: none. Key controls and defaults: Strength (0.075), Speed (0.32), Spread (0.45).

Slow three-axis drift with a still Reduced Motion alternative. Start with <softSpatialFloat>AMBIENT</softSpatialFloat> and tune it in Preview.

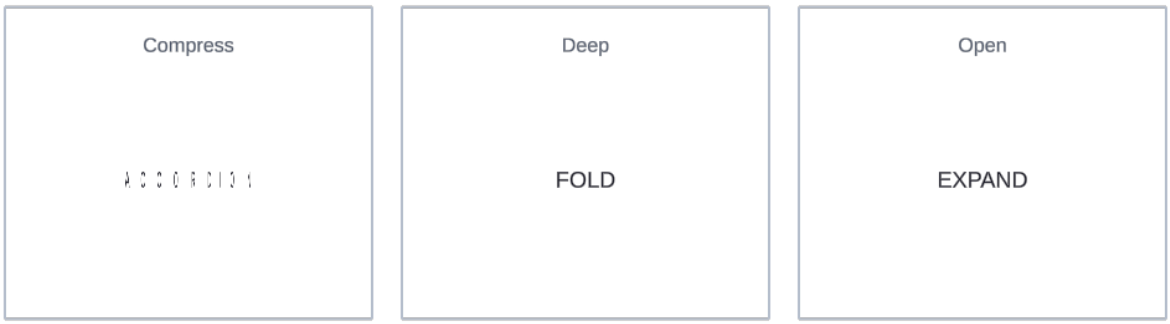


Soft Spatial Float: three useful starting points rendered by Text Studio.

Accordion

Canonical tag: <accordion>. Search aliases: none. Key controls and defaults: Compression (0.72), Depth (0.28).

Alternating width and depth compression before opening into place. Start with <accordion>ACCORDION</accordion> and tune it in Preview.



Accordion: three useful starting points rendered by Text Studio.

Orbit

Canonical tag: <orbit>. Search aliases: none. Key controls and defaults: Radius (0.12), Speed (0.55), Spread (0.85), Face Motion (true).

Letters orbit their authored positions while remaining editable. Start with <orbit>ORBIT</orbit> and tune it in Preview.

Orbit: three useful starting points rendered by Text Studio.