FxChain: Documentation

Welcome to the documentation and thanks again for your purchase!

Dependencies

- **Unity 6.0 LTS** (minimum compatible version)
- **Unity Splines** (optional) - install through the package manager
- **Timeline** (optional) - install through the package manager for the FxChain Timeline track.
- **Visual Effect Graph** (optional) - for the VFX Control component.
- **TextMeshPro / Unity UI** (optional) - for the Text and UI Animate components.
- **Burst** (optional) - Mass Repeater updates are Burst-compiled when it's installed.
- **Render Pipeline:** FxChain works with the Built-in Render Pipeline, URP and HDRP. The demo scenes and the Decal component need URP, and Volume Weight needs URP or HDRP.

What's New in v3.0

FxChain 3.0 adds playback control from scripts and Timeline, a Chain Timeline window to scrub effects, fourteen new components, new Repeater arrangements and many editor improvements. Existing effects keep playing exactly as they did in 2.0.

- **Chain Timeline Window:** Every component in a chain as a bar. Drag and scrub to preview any moment of the animation.
- **Scene View Playback Overlay:** Play, pause, stop and scrub the selected chain without leaving the Scene view.
- **Scripting API:** Play, PlayAsync, Pause, Resume, Stop, Seek, time modes, speed and events (see Playback from Scripts).
- **Looping:** Loop forever, ping-pong and restart when complete.
- **Timeline Track:** Play and scrub chains from Unity Timeline cutscenes.
- **14 New Components:** Light, Camera Shake, Event, Particle Control, UI Animate, Text, Branch, Look At, Blend Shapes, Physics Impulse, Volume Weight, Decal, VFX Control and the Mass Repeater.
- **Repeater:** Grid, Volume and Mesh Surface arrangements, Recycle Spawns and Max Active Spawns.
- **Existing Components:** Animate shake, per-axis curves and shortest-path rotation, Scaling per axis, Trail start delay and ribbon, Material Properties keywords, flipbooks, texture scrolling and global properties, Sprite Sheet Control billboarding, and audio variations.
- **Editor Tools:** A Getting Started guide, Chain Doctor, Chain Templates, Scene view handles, UI Toolkit inspectors with a timing bar, colour-coded components, Add Child Component, Preferences, Unity Search and an AI Assistant menu that adds the AI guide to Claude (see Editor Tools).
- **Demo Scenes:** Eleven new genre scenes with 44 effects, each also a ready-made template.

Upgrading from v2.0

- **Back Up First:** Back up your project, then import 3.0 over 2.0. Unity 6.0 or newer is required.
- **Your Effects Play as Before:** Every component made with 2.0 keeps v2.0 behaviour, so existing effects look exactly the same.
- **New Components Use v3.0 Behaviour:** Components you add from now on include the v3.0 fixes.
- **Upgrade Chain to v3.0 Behaviour:** To move an existing chain over, select its chain root and click this button (Undo is available). Check the effect afterwards, as the fixes can change how it looks.
  - **The Fixes:** Random spawn timings vary properly, spawns are reproducible from the Random Seed, the Min frame rate slows the animation instead of jumping, Material Index and Apply to All are respected at runtime, every sprite sheet plays in Sequential order, paths on rotated Line Renderers are oriented correctly, trails follow FxChain time, Scaling multiplies the object's own scale and can overshoot, and every play starts from the same pose.

- **Demo Folders:** The demo scenes are now URP only and sit directly in FxChain/ExampleScenes. An Asset Store update doesn't delete files, so remove 2.0's BuiltIn_RenderPipeline and HDRP_RenderPipeLine demo folders after updating.
- **Renamed Options:** Many inspector labels have clearer names in 3.0. Some sections below use the 2.0 names; see Renamed Options under Editor Tools for the list.

Core Concepts

Understanding these core ideas is key to mastering FxChain:

1. **Animation Chains:** An FxChain animation is built by linking specialized FxChain components (like Animate, Repeater, etc.) together. You add these components to GameObjects just like any other Unity component.
2. **Linking Components (Next In Chain):** Each FxChain component has a Next In Chain list in its Inspector. This list defines which component(s) will execute *after* the current one finishes. This creates the sequence or chain of animation. You can add to this list manually or use the Add Child Component button to quickly insert new FxChain components as children in the chain.
3. **Chain Hierarchy vs. Scene Hierarchy:** The animation chain's structure (parent/child relationships defined by the Next In Chain lists) is *independent* of the GameObject hierarchy in the Unity Hierarchy window. Components don't need to be nested physically to be linked in the animation chain.
4. **The Chain Root:** Every animation sequence starts from a designated Chain Root component. This is typically the first component in your sequence. You initiate the entire animation by triggering this root component.
5. **Timing Propagation:** Each component has Initial Pause and Duration settings. Crucially, timing information flows *down* the chain. Child components can automatically factor in their parent's timing (e.g., wait for the parent to finish), allowing for easy synchronization without manual calculations.
6. **Editor Preview:** Test your entire animation sequence directly in the Scene view using the Play Animation In Editor button on any FxChain component, or the playback overlay in the Scene view, without entering Play mode.
7. **Seeing the Whole Chain:** The Chain Timeline window and the timing bar in every component's Timing section show when each component plays. Drag across them to scrub the preview.
8. **Playing from Scripts:** Call Play(), PlayAsync() or Stop() on any component of a chain to control it from your own code (see Playback from Scripts).

Locating FxChain Components and Tools

- **Main Menu bar:** Components and tools can be found under Tools/FxChain.
- **Hierarchy Window:** Click the **+** Create dropdown button → FxChain for components.
- **Adding Directly to GameObjects:** Click the Add Component button and search for FxChain.
- **New Chain Roots:** A component added with Add Component goes straight under the Transform, and a component that starts a chain of its own lasts 1 second.
- **Add Child Component:** In a chain, the Add Child Component button in the Next In Chain section opens the components as buttons grouped by what they do. Click one to add it as a child in the chain.
- **Window Menu:** Window → FxChain opens the Getting Started guide, the Welcome window, the Chain Timeline, the Chain Doctor and Chain Templates.
- **AI Assistant:** Tools → FxChain → AI Assistant adds the AI guide to Claude Code or the Claude app, or copies it for other assistants (see Editor Tools).
- **Unity Search:** Type fxc: in Unity's Search window to list the FxChain components in the open scenes.
- **Preferences:** Edit → Preferences → FxChain holds the editor-wide options.

Component Overviews

Due to FxChain's procedural nature, sequences are built by combining modular components. You can think of them like lego blocks, stack them up, swap them in and out, mix up their order, play and see what you can create!

Animate

- Animates Position, Rotation, and Scale towards Set Destination Transform values.
- Supports World/Local space transformations independently.
- Applies Randomization Ranges to destination values (useful with Repeaters).
- Provides toggles for quick visual preview of the target transform.
- Ability to reverse the transform - starting at the target transform and moving towards the initial transform.
- Provides multiple easing curves for an array of parameters for fine-tuned control over movement. Some additional easing types are also included (Elastic, Back, Bounce).
- Optionally moves towards a dynamic target object (additive, blendable motion).
- Orients object towards a target direction, object, or its own trajectory.
- Alternative Follow Spline Mode: Animates along a Spline Container or Line Renderer path with additional options to align to spline, look ahead, banking, easing and offset control.
- Shake: a noise layer on position and rotation with an envelope over the animation.
- Per-axis easing curves and shortest-path rotation.
- Scene view handles: drag the end point to set the movement.

Repeater

- Spawns multiple instances of items from the Next in Chain list.
- Controls timing per spawn (delay/duration, random/sequential).
- Sets base spawn location/orientation via a birth transform object.
- Optionally attaches spawns to follow the birth transform object's motion.
- Arranges spawns: Point, Line, Arc, Mesh Vertices, Line Renderer points, Unity Splines.
- Can orient spawns to surface normals (Mesh/Spline/Line) or aim towards a target.
- Options to add additional randomization to individual spawn position, rotation, and scale.
- As well as progressive incremental changes to individual spawn position, rotation, and scale.
- Visualizes spawn locations and orientations in the Scene view.
- Ability to Preload Spawns: Whilst building sequences spawns are created when called for but once you're happy with your build you can enable preloading to avoid the slowdown associated with instantiation.
- Grid, Volume (box, sphere or disc) and Mesh Surface arrangements.
- Recycle Spawns: reuses finished spawns instead of creating and destroying them.
- Max Active Spawns: skips new spawns while that many are out, like a particle system at its maximum.

SpriteSheet Control (SSC)

- Animates sequences using pre-processed SpriteSheet scriptable object assets (setup via helper window).
- Can hold multiple SpriteSheets with controls for selection and play order: All / One & Sequential / Random.
- Ability to scrub through or preview SpriteSheet animations.
- Controls Playspeed FPS, looping, random start frame.
- Can automatically set Duration based on sheet length & FPS.
- Manages sprite scale (consistent visual size) and simulates anchor points.
- Option to apply random initial Z rotation.
- Animates Color and Emission over time using Gradients (alpha for fades).
- Supports inheriting materials passed down the chain.
- Can dynamically adjust Sorting Layer/Order based on world position or camera distance.
- Billboard: faces the camera, or stays upright and turns around Y only.
- The setup tool also accepts a Sprite Atlas.

Scaling

- Animates object scale using a precise Animation Curve.
- Remaps the curve's 0-1 output range to custom Min/Max scale multipliers.
- Provides direct, focused control over scale transitions.
- Per Axis: X, Y and Z can follow their own curves (squash and stretch).
- With v3.0 behaviour it multiplies the object's own scale, and its curves can overshoot.

Trail

- Creates dynamic trail effects using the unity LineRenderer component with time or distance-based modes:
- **Time Mode:** Removes points after specified duration.
- **Distance Mode:** Maintains trail within specified length.
- **Customizable trail appearance:**
  - Width control using animation curve and base multiplier.
  - Color and transparency using a gradient.
  - Width can scale with trail length/fullness.
- **Advanced trail behavior:**
  - Optional trail shrinking when object stops moving.
  - Automatic cleanup on animation start/stop.
  - Extends animation duration until movement completely stops.
- **Start Delay and Ribbon:** Start drawing part way through, or let the width follow the object's Z axis so it twists with it.
- **Performance optimization:**
  - Configurable update frequency.
  - Minimum distance threshold for new points.
  - Efficient point management and removal.

Material Properties

- Dynamically animates material properties over time using the Unity Material Property Block system.
- Supports multiple renderer types: MeshRenderer, SkinnedMeshRenderer, SpriteRenderer.
- Animates base color using a customizable gradient with alpha support.
- Controls emission color and intensity with separate gradient and power multiplier.
- Can target specific material slots or affect all materials on a renderer.
- Supports custom property animation:
  - Float values with animation curves.
  - Color values with gradients.
  - Vector values with linear interpolation.
- For SpriteRenderers: Optional shadow casting control.
- Changes persist only during animation, reverting to original material state after completion.
- A property picker listing the shader's real properties.
- Keyword, Flipbook and Texture Scroll properties, and Float, Color and Vector properties that can be set globally.

Parent Link

- Acts as a null/passthrough link in the chain, performing no animation itself.
- Useful for creating timing delays or branching points without adding animation behaviour.
- Helps organize complex chain structures.

Light

- Animates a Light's intensity, colour and range with curves over its duration.
- Random flicker for fire, faulty lights and sparks.
- Can add a Point Light when the object has none.

Camera Shake

- Shakes a camera's position and rotation with smooth noise.
- A strength envelope over the duration, and fall-off with distance.
- Several shakes add together, and the camera's own movement is kept (Play mode only).

Event

- Switches objects on or off and sets an Animator parameter when it starts.
- Calls On Start and On Complete UnityEvents in time with the chain.

Particle Control

- Plays, stops, clears or bursts a Particle System in time with the chain.
- Scales its emission rate and simulation speed over the duration, and previews in the editor.

UI Animate

- Moves, resizes, rotates and scales UI elements, to an offset or from it back home.
- Random -/+ per play, separate rotation and scale curves, and a shake.
- Fades a Canvas Group and tints an Image or Text.

Text

- TextMeshPro typewriter reveal with a per-character fade-in, and a wave.

Branch

- Plays only some of its Next In Chain children each time: a random few, one by weight, the next in order, or the one passed in from code.

Look At

- Turns the object to face a target, blended in and out by a weight curve.

Blend Shapes

- Drives Skinned Mesh Renderer blend-shape weights with curves.

Physics Impulse

- Pushes rigidbodies with a directional force or an explosion, plus torque, when it starts (Play mode only).

Volume Weight

- Fades a post-processing Volume in and out: hit flashes, chromatic pulses, slow-motion tints.

Decal

- Fades and grows a URP Decal Projector: scorch marks, impact rings, footprints.

VFX Control

- Sends events to a Visual Effect Graph at its start and end, and drives one exposed float with a curve.

Mass Repeater

- Draws thousands of copies of one mesh with GPU instancing, without creating GameObjects: sparks, debris, confetti, swarms.
- Emission shapes, outward or fixed direction with spread, speed, drag and gravity, spin or align to velocity, and size over life.
- Scrubbing and Seek show exactly what plays.

Component Shared Functionality

All FxChain components share some common UI elements in the Inspector.

- **Chain Position Indicator:** A number visually showing the component's depth in the chain. Note that scenes can contain multiple chains.
- **Play Animation in Editor Button:** Previews the entire animation chain starting from the Chain Root without having to enter play mode.
- **Timeline Button:** Opens the Chain Timeline window for this chain (see Editor Tools).
- **Component Colours:** Each component type has its own colour, shown as a tint and a stripe in the Hierarchy, in the Chain Timeline, the Next In Chain list and the add buttons. Parent Link and Branch are grey.

Next In Chain List

This is how items in the chain connect regardless of relationships in the hierarchy window.

- Displays the components executed immediately after this one.
- Simply drag / drop other FxChain components to insert them as child components in the chain.
- Additional buttons assist with removing items and altering their order.
- **View Chain Parent Button:** Click to ping the parent component. This can be handy when a chain gets complex or you have multiple chains present in the current scene.
- Provides buttons to automatically create and link new FxChain components (Animate, Repeater, etc.) as children in the chain.
- **Add Child Component:** A full-width button that opens the components as buttons grouped by what they do: Move, Spawn, Look, Light / Camera / Post, Particles / VFX, UI / Text and Flow. Click one to add it as a child in the chain; the buttons close again.
- **Child Timing:** The list can be reordered by dragging, and each child shows its colour and its timing.
- **Add Child Geometry:** An Animate, Scaling or Look At with nothing to show (no renderer on it or under it, as with a new child) shows *Nothing to show yet* with Sphere, Cube and Quad buttons. Each adds a small placeholder child you can swap for your own model.

Timings Section - Initial Pause

A delay before this component's animation starts.

- **Add Parent's Initial Pause (Default: On):** Adds the parent's delay. Disable to potentially start simultaneously with or even before the parent.
- **Add Parent's Duration (Default: On):** Waits for the parent's animation duration to complete before starting. This quickly allows sequential timing. Note that you could enable this and then set the initial pause value to a negative number to add an overlap in timings.

Timings Section - Duration

How long this component's own animation effect takes.

- **Add Parent's Duration:** Makes this component's duration equal to its parent's. Useful for effects that should run concurrently and finish together.

Timings Section - Proximity Override:

- **Note:** This is different from the Proximity trigger mode (set on the Chain Root). The trigger mode controls when the *entire chain* starts, while Proximity Override controls when *this specific component* starts within an already-running chain.
- **Enabled:** The initial pause will be overridden and this component will only start once in range of a specified object.
- **Proximity Target:** Select the object type that this script will use in the distance check.

- **Trigger Distance:** The distance at which this component will trigger when near the Proximity Target.

Timings Section - Loop:

- **Enabled:** This component's animation will loop. So if the Duration is set to 2 seconds and the Loop Count set to 3, total Duration will be 6 seconds. Note that this doesn't affect other components, in the previous scenario a child component, if referencing the duration will still read 2 seconds.
- **Loop Count:** The number of times to loop the animation.
- **Offset Transform:** If enabled, each loop of the animation continues from where the last loop ended, rather than resetting to the original start position.
- **Forever:** Loops until the chain is stopped (not available on Repeaters).
- **Ping-Pong:** Plays forward, then back, on alternate loops.

Timings Section - Behavior Summary:

A sentence summarizes the calculated timing (e.g., "*Waits X seconds, then animates for Y seconds*").

Timings Section - Timing Bar

A bar under the timing settings shows where this component plays in the whole chain:

- **Chain Time:** The bar runs from 0 to the end of the chain, with the end time shown on the right.
- **This Component:** Its bar shows when it plays. A lighter part shows start times spread across spawns, and a Repeater's spawn window is outlined.
- **Chain Parent:** A thin line beneath shows when its chain parent plays.
- **Click:** Opens the Chain Timeline.

Other Section (Common Options)

- **Random Seed:** Sets the seed for any randomization used within this component.
- **Ignore Parent Orientation:** If the parent chain link has a rotation, enabling this inserts an intermediary game object that resets the orientation for this component and its children. Useful when spawned by a Repeater with rotation values.
- **Rename Includes Script Name:** When renaming a script's game object, some information is automatically included, you can optionally include the component type.
  - **Enabled:** "[*ChainPosition*]_*ScriptName*_*CustomName*"
  - **Disabled:** "[*ChainPosition*]_*CustomName*"
- **Limit Decimal Places:** Adjusts UI precision for sliders and fields.
- **Hide TIPS:** Toggles inline help text within the Inspector.
- **Scale Slider Ranges:** Adjusts the min/max limits for UI sliders.
- **Hide Movement Gizmo when Previewing:** Toggles visibility of the animation path gizmo during editor preview.
- **Use Global Section Toggles:** Controls the visibility behaviour of the section dropdowns.
  - **Enabled:** Section open / closed status will be shared across all FxChain components.
  - **Disabled:** Each FxChain component will have its own memory of sections being open or closed.
- **Destroy Object Batch Size:** When destroying objects, this number limits the amount of objects destroyed each frame. This can be very beneficial when using Repeaters with a lot of spawns as it prevents frame dips associated with garbage collection.
- **Moved in v3.0:** Limit Decimal Places, Hide TIPS, Scale Slider Ranges, Hide Movement Gizmo when Previewing, Use Global Section Toggles and Rename Includes Script Name are now in Edit → Preferences → FxChain (the FxChain Editor Preferences button opens it), and Destroy Object Batch Size is now Destroy Objects Per Frame in Chain Root Options.

On Animation Start (External Script Trigger)

Here you can trigger your own public scripts / methods using the FxChain timing system.

Audio Clip

Insert an audio file here to play it when this component's animation begins.

- **Note:** This will automatically add an Audio Source component to the GameObject if it doesn't already have one.
- **Stop audio when animation completes:** Uncheck if the audio clip's duration exceeds the total duration of all elements in the chain.
- **Variations:** Each play picks the Audio Clip or one of these clips at random.
- **Random Pitch / Random Volume:** Each play picks its pitch and volume between Min and Max, from the Random Seed.
- **Pitch / Volume Over Time:** Multiplies this play's pitch and volume by curves over the duration.
- **Set 3D Sound:** Sets the Audio Source's spatial blend, min and max distance and rolloff.
- **Pooled One-Shot (outlives this object):** Plays the clip on a pooled Audio Source at this object's position (Play mode), so it keeps playing when the object is destroyed or recycled and sounds can overlap. It copies this object's Audio Source settings (mixer group...), and Stop Audio when Animation Completes doesn't apply to it.

Video Clip

Insert a video file here to play it when this component's animation begins.

- **Note:** This will automatically add a Video Player component to the GameObject if it doesn't already have one.
- **Stop video when animation completes:** Uncheck if the video clip's duration exceeds the total duration of all elements in the chain.

Video Clip - Additional Setup Steps

FxChain will automatically add a Video Player component and add specified video to it but some additional manual setup steps are required to view it in the scene.

- **Add a Canvas to your scene:** Hierarchy → UI → Canvas
- **Add a Raw Image as a child of the Canvas:** Hierarchy → UI → RawImage
- **Create a Render Texture:** Project → Create → Render Texture
- **Assign to Video Player:** Assign the Render texture to the Target Texture field of the video player.
- **Assign to Raw Image:** Assign the Render texture to the Texture field of the RawImage.
- **Shortcut:** The Show Video on a UI RawImage button in the Video section does all of the above in one step: it creates a Render Texture sized to the clip and sets up the Video Player and a RawImage.

Video Clip file type notes: If you require alpha channel you can use the WebM format with vp8. You can convert existing video files to this format using **Shutter Encoder**: https://www.shutterencoder.com - Just make sure to check "Enable Alpha Channel" under Advanced Features.

Components: Animate

Purpose: Moves, rotates, and scales the GameObject over time.

Set Destination Transform

- **Target Transform:** Define target position, rotation, and scale values using sliders.
  - **Use Data toggles:** These relate to starting the animation via a method call (see Starting Chains from Scripts (Manual Triggers) for additional info). It enables you to pass in dynamic values for pos/rot/scl and their random equivalents.
- **Randomization:** Apply randomization ranges to position, rotation, and scale – great for creating variations when used with a Repeater.

- **Shortest Path (no multi-turn spins):** Under Rotation's Random -/+. Rotates between the same start and end orientations along the shortest way instead of through the Euler angles, so there are no multi-turn spins or long ways round.
- **Space:** Choose between World or Local space transformations independently for movement and rotation. Using world position can be useful if a parent object has been scaled.
- **Preview / End at Initial Transform:** *(position / rotation / scale)* Enabling these toggles provides a dual functionality.
  - **Temporary Enable:** Provides a quick way to preview the end result of the transform animation.
  - **Permanent Enable:** This will fully invert the transformation animation. When playing the animation back, either through the Editor or Runtime, it will start at the destination transform and end at the initial transformation.
  - **Note:** Be wary when reparenting the GameObject when this setting is enabled. It will cause an unexpected change to the transform; To do so safely, first disable the toggles and then re-enable after you've moved the object in the hierarchy.
- **Easing:** Apply separate, nuanced motion curves using the Easing Types list (see below) or custom curves for Position, Rotation, and Scale transformations independently.
- **Easing -/+ toggle**: This shifts the curve range from [0 - 1] to [-1 to 1]. This allows for spring-like animations.
- **Per Axis:** With Custom Easing Curve, X, Y and Z of the position, rotation or scale can each follow their own curve (rise first, drift sideways later). They start as copies of the curve.
- **Use Same Type As Position:** By default Rotation and Scaling follow the position's easing. Set their easing to Custom Easing Curve to give them a curve of their own.

Target Object

- **Target:** Specify a target GameObject using the Object Types list (see below) to move towards dynamically during the animation.
- **Additive Motion:** This movement is calculated *in addition* to the primary Set Destination Transform animation.
- **Blending:** You can smoothly blend *between* the Set Destination Transform motion and the Target Object motion over the component's duration.
- **Easing Controls:** Uses two distinct Easing Types controls:
  - One defines the interpolation for the movement *towards* the Target Object.
  - Another defines how the blend *between* the Set Destination Transform and the Target Object motion occurs over time using a blending curve.

Orientation

- **Facing Direction:** Orient the GameObject's local forward (Z-axis) towards a specified target using the Object Types list.
- **Up Direction:** Orient the GameObject's local up (Y-axis) towards a specified target using the Object Types list.
- **Align to Movement:** Enable Orient Along Trajectory to automatically align the GameObject's forward direction to its path of movement.
- **Flip:** Independent toggles allow you to flip the final Facing and Up directions if needed.

Shake

A noise layer on top of the animated position and rotation, for wobbles, hits and rumbles.

- **Enable:** The toggle in the Shake section's header.
- **Position Amount:** How far the object moves on each axis.
- **Rotation Amount:** How far it turns on each axis, in degrees.
- **Frequency:** How fast the shake moves. Low = a slow sway, high = a rattle.
- **Envelope:** The amount over the animation (0-1 time). Ending at 0 leaves no offset when the animation completes.
- **Random Seed:** Each seed gives a different shake, and Repeater spawns get their own seeds.

Scene View Handles

- **End Point:** While the Animate is selected, drag its end point in the Scene view to set the movement. The box around it shows the random range.
- **Facing and Shake:** The Scene view also shows which way the object will face and its shake range.
- **Follow Path From / To:** In Follow Path mode, drag the From and To points along the path.
- **Proximity:** A sphere shows the proximity trigger distance.

Animate Follow Spline (Alternative Mode)

Purpose: Moves and orients an object along a spline.

- **Activation:** Enable this mode via the button to animate the GameObject along a path instead of using the standard transform animation.
- **Path Source:** Specify a GameObject that has either a Line Renderer or a Unity Spline Container component. FxChain will read the points or spline data from this source.
- **From / To:** Allows you to specify a section of the spline to animate along rather than the full length.
- **Playback:** Options for Looping, Reverse, and Ping Pong playback along the path.
- **Easing:** Control the traversal speed along the path using a specific Easing Type or a custom curve.
- **Offset:** Choose whether to Add Local Transform (adds the GameObject's initial local position/rotation as an offset to the spline path) or follow the spline path precisely.

Alignment & Orientation

- **Orient Along Path:** to automatically align the GameObject's forward direction.
- **Look Ahead %***:* Controls how far ahead on the path (as a percentage of total path length) the object looks for orientation calculation. A value of 0 uses the current tangent; a positive value looks ahead, causing the object to anticipate turns.
- **Up Direction:** Sets the object's Y-axis orientation, using the standard Object Types list.

Banking (Roll)

- **Enable Banking:** to make the object realistically roll into turns.
- **Max Tilt:** Limits the maximum roll angle (degrees).
- **Turning Angle Range:** The sharpness of the turn is evaluated and banking is applied linearly between the lower and upper limits.
  - **Lower value:** When the turn angle is less than the lower value, no banking is applied (deadzone).
  - **Upper value:** The Max Tilt value is linearly interpolated between the lower and upper value.
- **Smooth Factor:** Controls how quickly the object transitions to its calculated target bank angle. Higher values are faster/snappier; lower values are smoother/more gradual.
- **Sensitivity:** Multiplier for how aggressively the object responds to path turns.
  - Values <1 mean less bank for a given turn.
  - >1 means more bank, up to the Max Tilt.

Components: Repeater

Purpose: Instantiates multiple copies ("spawns") of the next component(s) in the chain.

Core Function

Creates multiple instances of the GameObject(s) holding the FxChain component(s) linked in the Repeater's Next In Chain list. Any child objects nested under the spawned component's GameObject in the Scene Hierarchy are also duplicated. Multiple Repeaters can be chained for complex, particle-system-like effects.

Spawn Timing (Overrides Standard Timings)

- The Repeater component's main Initial Pause value acts as a delay *before* the spawning process begins.
- The Initial Pause and Duration fields within this Timings section apply *to each individual spawn* created by the Repeater.
- **Timing Mode (Dropdown):** Determines how the Initial Pause and Duration values are assigned per spawn:
  - **Sequential:** The value is interpolated between the Min and Max based on the spawn's index (order of creation). The first spawn gets the Min value, the last spawn gets the Max value.
  - **Random:** Each spawn receives a random value between the specified Min and Max.
  - **Distance Based (Initial Pause Only):**Calculates the *Spawn's Distance* from the *Distance Reference Object*.
    - Closest Spawn → Initial Pause Min
    - Furthest Spawn → Initial Pause Max
    - Requires setting the Distance Reference Object option (only viewable when using this Timing Mode).
- **Min/Max Ranges:** Define the range for Initial Pause and Duration per spawn used by the selected Timing Mode.
- **Subtract Own Initial Pause (Duration Only):** Useful with ranged initial pauses (especially Sequential) to make all spawns *finish* their animations simultaneously, rather than starting simultaneously.

Spawn Configuration

- **Birth Count:** Number of spawns to create (can be overridden by some arrangement modes).
- **Use Data Toggle:** This relates to starting the animation using a method call and passing in a dynamic birth count value. For example if it's a damage animation, you could pass in the number of times the character has been hit.
- **Spawn Selection Behaviour:** If multiple components are in the Next In Chain list, choose whether to cycle through them sequentially for each new spawn, or select one randomly per spawn.

Birth Transform

- **Birth Transform Object:** Specify an object (Object Types list) whose transform dictates the base location/orientation for spawning.
- **Inherit Object's Motion (Attach Spawns):**
  - **Enabled:** Spawns become children of the Birth Transform Object (or a proxy created by the Repeater) and follow its movement.
  - **Disabled:** Spawns only use its position/rotation at the moment of birth and do not follow it afterwards.

Spawn Destruction

- **Destroy spawns when their animation is complete:** The spawn and its subsequent child scripts will be checked for when their animation completes. Once all animations are complete they will be destroyed.
- Also note that this process uses the staggered destruction process to avoid performance issues.
- **Max Active Spawns (0 = no limit):** Above 0, a spawn that's due while this many are out (made and not finished) is skipped, like a particle system at its maximum. With Destroy Spawns When Complete off, finished spawns stay, so it limits the spawns of each play.
- **Recycle Spawns (reuse finished ones):** Finished spawns are reset and kept, and new spawns reuse them instead of being created and destroyed again. Effects that play often spend 50-75% less time creating spawns and make less garbage, and the kept spawns stay in memory until the scene unloads. On for new Repeaters, and not available with preloading. Your own scripts on a spawned object run Awake / Start once per kept spawn and their changes stay on it: turn recycling off if they need a fresh object (Chain Doctor points this out).

Arrangement

Birth Arrangement (Dropdown): Defines how spawns are positioned relative to the Birth Transform Object:

- **Point:** All spawns originate at the exact same point.
- **Line:** Spawns are distributed along a line (specify Width, Randomness).
- **Arc:** Spawns are distributed along an arc/circle (specify Axis, Radius, Direction, Range, Randomness).

- **Mesh Points:** Spawns are placed at the vertex positions of a Mesh Filter on the Birth Transform Object.
- **Line Renderer:** Spawns are placed at the points of a Line Renderer on the Birth Transform Object.
- **Unity Splines:** Spawns are placed along a Unity Spline Container on the Birth Transform Object.
- **Grid:** X × Y × Z spawns on a lattice centred on the birth position, set with Count (X, Y, Z) and Spacing. The count replaces Birth Count. With Sequential timing, X fills first, then Y, then Z.
- **Volume:** Birth Count spawns at random inside a box, a sphere or a flat disc (XZ), spread evenly through the shape and set with Shape and Size. The positions come from the Random Seed.
- **Mesh Surface:** Birth Count spawns at random points on the mesh's triangles, weighted by area, with interpolated normals (Mesh Points uses the vertices instead).

Arrangement - Additional Options for Mesh/Line/Spline

- **Override Birth Count:** If enabled, the number of spawns created matches the number of vertices/points in the source component (Mesh Filter, Line Renderer, Spline Container). If disabled for Line Renderer or Unity Splines, the specified Birth Count number of spawns will be distributed evenly along the path(s).
- **Ignore Overlapping Points (Mesh Points Only):** Prevents spawning multiple instances at the exact same vertex location.
- **Orient to Normals:** Aligns each spawn's forward (Z-axis) direction with the normal vector of the mesh vertex or the tangent of the line/spline at the spawn point. Disables standard Spawn Aim & Orientation below.
- **Up direction (Object Types list):** Sets the up (Y-axis) orientation for spawns, used in conjunction with Orient to Normals.
- **Skinned Meshes:** Mesh Points and Mesh Surface also work with a Skinned Mesh Renderer, using its pose when the chain plays.
- **Read/Write:** For player builds the mesh needs Read/Write enabled in its import settings. Chain Doctor warns when it isn't.

Spawn Additional Transforms

Spawn Positions, Rotations and Scaling can be further manipulated on an individual basis via either the progressive or random transform values. These will apply additional offsets to the initial Position, Rotation, and Scale of each individual spawn.

Spawn Aim & Orientation

Similar to the Animate component's Orientation section, but applies *initial* orientation to spawns as they are created. This section is disabled if Orient to Normals is active in the arrangement settings.

- **Facing Direction:** Orient each spawns' local forward (Z-axis) towards a specified target using the Object Types list.
- **Up Direction:** Orient each spawns' local up (Y-axis) towards a specified target using the Object Types list.
- **Flip:** Independent toggles allow you to flip the final Facing and Up directions if needed.

Scene Visualization

- **Show Spawn Reference Objects:** Display temporary primitive shapes (e.g., cubes) in the Scene view to visualize where spawns will appear based on the current settings. Only visible in the Editor.
- **Show Spawn Trajectory Lines:** Draw lines from the reference objects indicating their initial forward (Z - blue) and up (Y - green) directions, helpful for visualizing orientation settings. Only visible in the Editor.
- **Draw as Gizmos (no objects):** Each spawn location is drawn as a light marker with an arrow along its forward direction instead of a visible object, shown while the Repeater is selected.

Components: Spritesheet Control

Purpose: Animates sequences of sprites from pre-configured Spritesheet assets.

Preparing Spritesheet Textures for Use with the Component

- Spritesheets must first be processed into a custom FxC_SpriteSheet Scriptable Object asset. This asset holds the individual sprite frames in the correct sequence for animation.
- Each Texture must be set up first with the following settings:
  - **Texture Type:** Sprite (2D and UI),
  - **Sprite Mode:** Multiple, and pre-sliced in Unity's Sprite Editor tool).

Setup (From Within the SSC Component)

- Click the Setup New Spritesheet button to open a helper window.
- The tool packages the slices into a SpriteSheet asset.
- It provides info on sprite count and save location as well as giving the following options:
  - **Maintain Aspect Ratio:** Controls whether the SpriteSheet preview image scales to the full size of the helper window.
  - **Insert Null Frame at Start / End:** Unity's Sprite Editor tool will remove empty frames from the image but they are helpful if you need the animation to start / end without content on screen.
  - **Rename:** Provides the option to rename the SpriteSheet Object. If unchecked it will simply use the pre-existing name of the image.
  - **Sprite Atlas:** The tool (Tools → FxChain → Setup New SpriteSheet Object) also accepts a Sprite Atlas as the source.

Selection Behaviour (Multiple SpriteSheets)

The component allows you to add multiple SpriteSheets. The following options determine how sheets are chosen, especially when spawned by a Repeater.

- **ALL + Sequential:** Plays all listed sheets in order.
- **ALL + Random:** Plays all listed sheets in a random order.
- **ONE + Sequential:** Each Repeater spawn gets the next sheet from the list in order.
- **ONE + Random:** Each Repeater spawn gets a random sheet from the list.

Preview Spritesheets

- Scrub through frames or play the sequence(s) in the Inspector.
- Shows current sheet index, frame number, and total frames.

Spritesheet Options

- **Set Duration from Spritesheet:** Automatically sets the component's Duration based on the total frame count of selected sheets and the Playspeed FPS, overriding the main Timings section Duration.
- **Playspeed FPS:** Playback speed in frames per second.
- **Random Start Frame:** Begin playback at a random frame within the sequence.
- **Loop Count:** Number of times to repeat the entire animation sequence.
- **Set Last Frame to Null:** Clears the Sprite Renderer's displayed sprite after the animation finishes.

Spritesheet Transform

- **Sprite Scale:** Applies a uniform scale multiplier. The actual object scale is calculated based on this multiplier, the sprite's dimensions, and its pixels per unit, ensuring consistent visual size across different sprites.

- **Sprite Anchor Point:** Simulates changing the sprite's anchor point (pivot) by adjusting the GameObject's local position relative to the selected anchor preset (e.g., Top-Left, Center, Bottom-Right), scale, and rotation.
- **Random Z Rotation:** Apply a random initial Z rotation (useful when spawned by a Repeater).
- **Billboard:** While playing, the sprite turns to face the camera (parallel to the view, so it doesn't skew at the screen edges). Upright only turns around world Y, for flames and smoke.

Color and Emission over Time

- **Change Color Over Time:** Allows you to change the color and opacity over time using the gradient editor.
- **Change Emission Over Time:** Allows you to change the emission color and opacity over time using the gradient editor. (Emission requires a material with an emission channel).
- **Emission Power:** The emission color will be multiplied by this value.
- **Inherit Gradient from Color:** If enabled, the emission gradient will inherit and match the color gradient.

Material to Use and Shadow Casting

- **Material:** Choose to use the material assigned to the Sprite Renderer component or inherit a material passed down the FxChain (see Starting Chains from Scripts (Manual Triggers) for additional info).
- **SpriteRenderer Shadow Casting:** Enables the sprite renderer to cast shadows. The material should also have shadow casting and alpha clipping enabled.

Material Property Names

- **Color Property Name:** The shader property name to animate (automatically set based on render pipeline):
  - **Standard Pipeline:** "_Color"
  - **URP / HDRP:** "_BaseColor"
- **Emission Property Name:** The shader property for emission (automatically set based on render pipeline):
  - **Standard/URP:** "_EmissionColor"
  - **HDRP:** "_EmissiveColor"
- **Sprite Tex Property Name:** The shader property name for the sprite texture (automatically set based on render pipeline):
  - **Default for Sprites:** “_MainTex"
  - **Standard Pipeline**: "_ColorMap"
  - **URP / HDRP:** "_BaseColorMap"

Dynamic Sorting

- Modify the Sprite Renderer's Sorting Layer and Order in Layer dynamically based on position.
- Choose a Threshold Axis (X, Y, Z world position, or Distance From Camera).
- Define different Sorting Layer/Order in Layer values for when the object is above or below the Threshold Value. Useful for ensuring correct render order for Repeater spawns at different depths or positions.

Components: Scaling

Purpose: Animates the GameObject's scale using a precise curve.

- Provides a standard Unity Curve Editor to define scale over the component's Duration.
- Remap the curve's output (default 0-1) to a custom Min and Max scale multiplier.
- Simple and direct control over scaling animations.
- **Per Axis (X / Y / Z curves):** X, Y and Z each follow their own curve (red, green, blue): squash and stretch, or growing one way first. Turning it on starts Y and Z as copies of the curve.
- **v3.0 Behaviour:** Scaling multiplies the object's own scale, so non-uniform scales are kept, and the curve can overshoot past 0-1 for bouncy pops.
- **Note:** The transform's scale values are overridden while it plays.

Components: Material Properties

Purpose: Animates material properties over time using Unity's Material Property Block system for efficient, non-destructive material changes.

Renderer Settings

- **Renderer Type:** Choose the type of renderer component to use:
  - MeshRenderer (with MeshFilter).
  - SkinnedMeshRenderer.
  - SpriteRenderer.
- **Material Index:** Select which material slot to modify when the renderer has multiple materials.
- **Apply to All Materials:** When enabled, changes affect all material slots on the renderer.
- **SpriteRenderer Shadow Casting:** For SpriteRenderer type only, enables shadow casting with proper material setup.

Color Over Time

- **Enable Color Animation:** Toggles color property animation using a gradient.
- **Color Gradient:** Define how the color changes over the animation duration, including alpha for transparency.

Emission Over Time

- **Enable Emission Animation:** Toggles emission property animation using a gradient.
- **Emission Gradient:** Define how the emission color changes over time.
- **Emission Power:** Multiplier for the emission color intensity.
- **Inherit Gradient from Color:** If enabled, the emission gradient will inherit and match the color gradient.

Material Property Names

- **Color Property Name:** The shader property name to animate (automatically set based on render pipeline):
  - **Standard Pipeline:** "_Color"
  - **URP / HDRP:** "_BaseColor"
- **Emission Property Name:** The shader property for emission (automatically set based on render pipeline):
  - **Standard / URP:** "_EmissionColor"
  - **HDRP:** "_EmissiveColor"

Custom Properties

Add and animate any custom shader properties:

- **Property Types:**
  - **Float:** Animated using a custom curve.
  - **Color:** Animated using a gradient.
  - **Vector:** Linear interpolation to target value.
  - **Keyword:** Turns a shader keyword (e.g. "_EMISSION") on while the curve is at 0.5 or above. While the chain plays the material slots in use are copies, so the material asset never changes.
  - **Flipbook:** Steps a texture's tiling and offset through a grid of frames (Columns / Rows, Frames, and Frames Per Second, where 0 spreads the frames over the duration). A GPU flipbook on any quad, with no sprite swaps.
  - **Texture Scroll:** Scrolls a texture's offset continuously across loops, set with Tiling and Scroll Speed (texture widths per second).
  - **Global:** For Float, Color and Vector: sets a global shader property read by every material that uses it, and puts the previous value back on reset.
- **Each Property Requires:**
  - **Property Name:** Exact shader property name (e.g., "_Metallic").

- **Value/Curve/Gradient** based on type.
- Properties can be added/removed during setup.
- **Property Picker:** The ▼ button beside the name lists the shader's properties of that type, and a warning shows when a typed name doesn't exist on the material.

Note: All material changes are temporary and revert when the animation completes. The original material remains unmodified.

Components: Trail

Purpose: Creates dynamic trail effects that follow object movement using LineRenderer.

Trail Length Mode

- **Time-based:** Removes points after specified duration.
  - **Trail Duration:** How long each point remains (seconds).
- **Distance-based:** Maintains trail within specified length.
  - **Trail Length:** Maximum distance the trail can extend.

Start Delay and Ribbon

- **Start Delay:** The trail starts drawing this many seconds into the component, from wherever the object is then.
- **Ribbon:** The trail's width follows the object's Z axis instead of facing the camera, so it twists as the object turns (blade swipes, ribbons).
- **FxChain Time (v3.0 behaviour):** Trails pause with the chain and respect the playback frame rate limits.

Trail Shrinking

- **Shrink Trail:** Enables shrinking behaviour to reduce the length of the trail.
  - **Start Shrink At:** The percentage of the animation duration at which the trail starts shrinking. Once the animation is complete, the trail will have disappeared.
- **Velocity Based Shrink:** When enabled, the trail shrinks based on the object's movement speed.
  - **Velocity Based Shrink Speed:** The speed at which the trail shrinks based on the object's movement speed. Higher values make the trail shrink faster.

Performance Settings

- **Minimum Distance:** Minimum movement required to add a new trail point.
- **Update Frequency:** How often the trail updates (Hz).
- **Clear Trail:** Options to remove trail points on animation start or end.

Trail Appearance

- **Width Control:**
  - **Base Width:** Overall width multiplier.
  - **Width Over Length:** Animation curve controlling width along trail (0 = start, 1 = end).
  - **Width Multiplied by Length:** When enabled, the trail's width scales based on the current trail length compared to the maximum trail length.

Trail Color and Emission Gradients

- **Color Gradient:** Controls color and transparency along trail length.
- **Emission Gradient:** Controls the emission color and transparency along the trail.
- **Emission Power:** The emission color will be multiplied by this value.
- **Inherit Gradient from Color:** If enabled, the emission gradient will inherit and match the color gradient.

- **Gradient Texture Steps:** The gradient textures are applied to the material as a generated texture. The texture is 1px high and the width is determined by this value. Higher values = more detailed gradient

Material Property Names

- **Color Property Name:** The shader property name to animate (automatically set based on render pipeline):
  - **Standard Pipeline:** "_Color"
  - **URP / HDRP:** "_BaseColor"
- **Emission Property Name:** The shader property for emission (automatically set based on render pipeline):
  - **Standard / URP:** "_EmissionColor"
  - **HDRP:** "_EmissiveColor"
- **Main Tex Property Name:** The shader property name for the main texture (automatically set based on render pipeline):
  - **Standard Pipeline**: "_ColorMap"
  - **URP / HDRP:** "_BaseColorMap"
- **Emission Map Property Name:** The shader property name for the emission texture (automatically set based on render pipeline):
  - **Standard Pipeline**: "_EmissionColorMap"
  - **URP / HDRP:** "_EmissiveColorMap"

Components: Parent Link

Purpose: Acts as a null or passthrough link in the chain.

- Performs no animation itself but respects timing.
- Useful for organizing complex chains, acting as a branching point or a delay node without adding other animation behaviour. Can link to multiple children in its Next In Chain list.

Components: Light

Purpose: Animates a Light's intensity, colour and range over the component's duration.

Flashes for explosions, glows for portals and power-ups, flickering torches. The Light's own values come back when the chain is reset.

Light Source

- **Target:** The Light to animate. Empty: the Light on this object.
- **Add a Point Light:** Shown when there's no Light to animate. Adds one to this object.

Intensity

- **Enable:** The toggle beside the Intensity title. Unticked, the Light keeps its own intensity.
- **Peak:** The intensity at curve value 1 (Intensity = Peak x the curve).
- **Over the duration:** A curve of the intensity from the start (left) to the end (right) of the duration.
- **Flicker:** How far the intensity can dip at random. 0 = steady, 1 = it can drop to nothing.
- **Speed:** How fast the flicker changes.

Colour

- **Over the duration:** The Light's colour along the gradient, from the start (left) to the end (right).

Range

- **Peak:** The range at curve value 1 (Range = Peak x the curve).
- **Curve:** The range over the duration.

Components: Camera Shake

Purpose: Shakes a camera with smooth noise over the component's duration.

Several shakes on one camera add together, and the camera's own movement (follow scripts, Cinemachine) is kept. It plays in Play mode only: editor previews don't move the Scene view camera.

Camera

- **Target:** The camera to shake. Empty: the main camera.

Shake

- **Strength:** Scales both amounts below (2 = twice as strong).
- **Position:** How far the camera moves on each axis, in local units at strength 1.
- **Rotation:** How far the camera turns on each axis, in degrees at strength 1.
- **Frequency:** How fast the shake moves. Low = a slow sway, high = a sharp rattle.

Envelope

- **Strength over the duration:** A curve of the strength. The default starts strong and dies away.

Distance

- **Fade Out At:** Above 0, the shake fades to nothing when the camera is this far from this object, so distant blasts shake less. 0 = the same everywhere.
- **Scene View:** While the component is selected, the Scene view shows how far the shake reaches.

Components: Event

Purpose: A timed node that switches objects, sets Animator parameters and calls your own methods in time with the chain.

When its animation starts (after its Initial Pause) it switches objects, sets an Animator parameter and calls On Start. When its duration ends it calls On Complete. Use it to start sounds, enable colliders, trigger animations or call your own scripts in sync with the effect.

Calls

- **On Start:** A UnityEvent called when the animation starts.
- **On Complete:** A UnityEvent called when the duration ends.

Switch Objects

- **On At Start:** Objects switched on when the animation starts.
- **Off At Start:** Objects switched off when the animation starts.

- **Restore On Reset:** Puts the switched objects back as they were when the chain is reset.

Animator

- **Target:** The Animator to set a parameter on.
- **Parameter:** The parameter's type: Trigger, Bool, Int or Float. None: no parameter is set.
- **Name:** The parameter's name, as in the Animator Controller.
- **Value:** The value it's set to. Bool: 0 = false, anything else = true. Int: rounded to a whole number.

Components: Particle Control

Purpose: Plays, stops or bursts a Particle System in time with the chain.

It also scales the system's emission rate and simulation speed over the component's duration. Editor previews simulate the system to the chain's time, so scrubbing shows the particles too.

Particle System

- **Target:** The Particle System to control. Empty: the one on this object.
- **Include Children:** Child Particle Systems are played, stopped and scaled too.

Playback

- **At Start:** What happens when the animation starts:
  - **Play:** Starts the system.
  - **Stop:** Stops the system.
  - **Stop And Clear:** Stops it and removes the live particles.
  - **Emit:** A single burst of Count particles.
  - **None:** Leaves it as it is.
- **Count:** Emit only: how many particles the burst makes.
- **At End:** What happens when the duration ends: Stop Emitting (live particles finish their lives), Stop And Clear, or None.

Emission Rate

- **Enable:** The toggle beside the title.
- **Curve:** Multiplies the system's own emission rate over the duration (1 = unchanged, 0 = none).

Simulation Speed

- **Enable:** The toggle beside the title.
- **Curve:** Multiplies the system's own simulation speed over the duration (below 1 = slow motion).

Components: UI Animate

Purpose: Animates a UI element: slide-ins, pops, fades, colour flashes, a shake on a wrong answer.

Works on anything with a RectTransform. Movement is relative to where the element is, and its own values come back when the chain is reset. For menus shown while the game is paused, set the chain root's Time to Unscaled.

Direction

- **To Offset:** Eases from where the element is to the offset (slide out, grow, turn).
- **From Offset:** Starts at the offset and eases back to where the element is (slide in, pop in). The element stays where you placed it in the layout.

Position

- **Enable:** The toggle beside the title.
- **Offset:** Pixels added to the anchored position (X, Y).
- **Random -/+:** Each play adds up to this either way on each axis. It comes from the Random Seed, so Repeater spawns differ.

Size

- **Offset:** Pixels added to the width and height (W, H).

Rotation

- **Degrees:** Turns the element around its pivot.
- **Random -/+:** Each play adds up to this either way.

Scale

- **At End / At Start:** Multiplies the element's own scale (1 = unchanged). The label follows the direction.
- **Random -/+:** Added to the X and Y multiplier, either way.

Easing

- **Position, Size, Rotation and Scale:** One curve for everything that moves (0 = the start, 1 = the end of the movement).
- **Separate Rotation and Scale Curves:** Rotation and Scale follow curves of their own, and the first curve drives Position and Size only (a pop that settles while the element slides).

Fade & Colour

- **Fade:** A Canvas Group's alpha over the duration (0 = invisible, 1 = fully shown). It fades the element and all its children.
- **Canvas Group:** Empty: the Canvas Group on this object, added when needed.
- **Colour:** The Image, Text or other Graphic's colour along a gradient, from the start (left) to the end (right).
- **Graphic:** Empty: the one on this object.
- **Note:** Fade and Colour always run from start to end: From Offset doesn't reverse them.

Shake

- **Enable:** The toggle in the Shake section's header.
- **Position:** How far the element moves on X and Y, in pixels.
- **Rotation:** How far it turns, in degrees.
- **Frequency:** How fast it shakes.
- **Strength over the duration:** The default starts strong and dies away.

Components: Text

Purpose: Animates a TextMeshPro text (UI or 3D) over the component's duration.

The text is shown in full again when the chain is reset. It needs TextMeshPro (the Unity UI package).

Text Object

- **Target:** The TextMeshPro text. Empty: the one on this object.

Typewriter

- **Enable:** The toggle beside the title.
- **Characters shown over the duration:** A curve of the fraction of characters shown (0 = none, 1 = all). The default finishes at 80% of the duration.
- **Fade-In:** Seconds each character takes to fade in after it appears. 0 = it pops in.

Wave

- **Enable:** The toggle beside the title.
- **Height:** How far the characters bob up and down.
- **Speed:** How fast the wave travels.
- **Spacing:** The phase step between neighbouring characters. Small = the text rises and falls together, large = a ripple.

Components: Branch

Purpose: Plays only some of its Next In Chain children each time the chain plays.

The others, and everything after them, are skipped and hidden for that play. Use it for combos, random outcomes, rare drops and results chosen by your game code.

Pick

- **Mode:** Which of the children play:
  - **Random:** A random few play each time.
  - **Weighted:** One plays, picked at random by weight. A child with weight 2 is picked twice as often as one with weight 1.
  - **Sequence:** The children take turns in list order: the first on the first play, the second on the next, and so on.
  - **From Chain Data:** Plays the child whose index is passed in when the chain is played: chain.Play(new ChainData(BranchPick: 2)). Without data it plays the first.
- **How Many:** Random only: how many of the children play each time.
- **Weights:** Weighted only: one per child, in list order. Missing weights count as 1.

Spawns and Code

- **Repeater Spawns:** Picks come from the Random Seed, so each spawn picks differently (in Sequence they take turns).
- **From Code:** branch.PickNext(index) sets the next play's pick in any mode, from a condition or your game's state.

Components: Look At

Purpose: Turns the object to face a target, blended over the component's duration.

It layers on top of rotation from other components, and the rotation comes back when the chain is reset.

Target

- **Object:** The object to face.
- **Up:** The world direction kept as up while facing (usually Y).

Weight

- **Over the duration:** How far it turns to the target (0 = its own rotation, 1 = facing the target).

Components: Blend Shapes

Purpose: Drives blend-shape weights over the component's duration: expressions, morphs, squash and stretch.

The renderer's own weights come back when the chain is reset.

Skinned Mesh

- **Target:** The Skinned Mesh Renderer. Empty: the one on this object.
- **Shapes:** The mesh's blend shapes, by name, to copy into the tracks.

Tracks

One track per blend shape to drive:

- **Shape Name:** The blend shape, as listed under Shapes. A warning shows when the mesh doesn't have it.
- **Max Weight:** The weight at curve value 1 (usually 100).
- **Weight Over Time:** The weight over the duration (x Max Weight).

Components: Physics Impulse

Purpose: Pushes rigidbodies when the component's animation starts.

Debris flying from a ground hit, objects knocked back by a blast. It plays in Play mode only (physics doesn't run in editor previews). While selected, the Scene view shows the push or the blast radius.

Force / Explosion

- **Force:** Pushes the bodies in one direction.
- **Explosion:** Pushes every body within the radius away from this object.

Bodies

- **Rigidbodies:** The bodies to push. Empty: for Force, the rigidbodies on this object and its children; for Explosion, every rigidbody within the radius.
- **Layers:** Explosion only: the layers it reaches.
- **Wake Kinematic Bodies:** Turns kinematic rigidbodies into simulated ones before pushing them (debris that sits still until it's hit).

Force

- **Force:** Explosion: the strength of the push. Force: its direction and strength (X, Y, Z).
- **Radius:** Explosion only: how far it reaches.
- **Upwards:** Explosion only: lifts the push, as Rigidbody.AddExplosionForce's upwards modifier.
- **Torque:** A spin added to each body (X, Y, Z).
- **Local Space:** Force and torque directions are in this object's space, so turning the object aims them.
- **Mode:** Unity's ForceMode. Impulse (the default) is a sudden push; Velocity Change is the same but ignores mass.

Variation

- **Randomness:** Each body's force varies by up to this fraction, stronger or weaker, from the Random Seed (0.2 = up to 20%).

Components: Volume Weight

Purpose: Fades a post-processing Volume in and out over the component's duration.

A white flash or chromatic pulse on impact, a tint during slow motion. The Volume's own weight comes back when the chain is reset. It needs URP or HDRP.

Volume

- **Target:** The Volume to fade. Empty: the Volume on this object. Use a local or global Volume with a profile of its own.

Weight

- **Peak:** The weight at curve value 1 (Weight = Peak x the curve).
- **Over the duration:** The weight over the duration (0 = no effect, 1 = full).

Components: Decal

Purpose: Fades and grows a URP Decal Projector over the component's duration.

Scorch marks, impact rings, footprints. The URP renderer needs the Decal renderer feature. The projector's own fade and size come back when the chain is reset.

Decal Projector

- **Target:** The Decal Projector. Empty: the one on this object.

Fade

- **Over the duration:** The projector's fade factor (0 = invisible, 1 = fully shown).

Size

- **Enable:** The toggle beside the title.
- **Curve:** Multiplies the projector's own size over the duration (1 = unchanged). The default starts small and grows.

Components: VFX Control

Purpose: Starts and stops a Visual Effect Graph in time with the chain.

It also drives one exposed float over the component's duration. It needs the Visual Effect Graph package.

Visual Effect

- **Target:** The Visual Effect. Empty: the one on this object.

Events

- **At Start:** The event sent when the animation starts (OnPlay starts most effects). Empty: none.
- **At End:** The event sent when the duration ends (OnStop stops most effects spawning). Empty: none.

Exposed Float

- **Name:** An exposed float in the graph's Blackboard: spawn rate, size, intensity... Empty: none.
- **Scale:** The value at curve value 1 (Value = Scale x the curve).
- **Curve:** The value over the duration.

Components: Mass Repeater

Purpose: Draws thousands of copies of one mesh with GPU instancing, without creating GameObjects.

Sparks, debris, confetti, swarms. Everything follows from the time and the Random Seed, so scrubbing and Seek show exactly what plays. Instances have no colliders, scripts or chains of their own: for spawns that play their own chain, use the Repeater.

Mesh & Material

- **Mesh:** The mesh every instance draws.
- **Material:** Its material, which needs GPU Instancing. Without it, a copy with it on is drawn, and a button turns it on for the material asset.

Emission

- **Count:** How many instances (1 to 100,000).
- **Emit Over:** The portion of the duration the instances are born over, evenly. 0 = all at once (a burst), 1 = across the whole duration.
- **Lifetime:** The portion of the duration each instance lives, between Min and Max at random. Every instance is gone by the end of the duration, so late ones live shorter.

Emission Shape

- **Point:** All from one point.
- **Sphere:** Inside a sphere of the Radius. Surface Only: on its surface.
- **Box:** Inside a box of the Size.
- **Circle:** On a flat circle in this object's X-Z plane. Edge Only: on its rim (rings and shockwaves).
- **Mesh Surface:** On a mesh's surface. Mesh Source: a Mesh Filter or Skinned Mesh Renderer (its pose when the chain plays); empty: this object. Builds need Read/Write on the mesh.

Motion

- **Direction Mode:** Outward: away from the shape (from its centre, along the surface normal on a mesh, every way from a point). Fixed: one direction for all.
- **Vector:** Fixed only: the direction, in this object's space.
- **Spread:** Degrees around the direction, at random. 0 = exactly along it, 180 = any way.
- **Units / Second:** Each instance's starting speed, between Min and Max at random.
- **Drag:** Slows the instances down over time.
- **Gravity:** Pulls them down, as a multiple of 9.81 in world space. Negative values lift them (smoke, embers).
- **Simulation Space:** Local: the instances move with this object. World: each stays where it was born as the object moves (sparks left behind a moving object).

Appearance

- **Orientation:** Fixed: as the mesh. Random: a random rotation each. Align to Velocity: facing the way it moves (sparks, streaks).
- **Spin:** Fixed and Random: degrees per second about a random axis, either way round, between Min and Max.
- **Stretch with Speed:** Align to Velocity: the length grows by this x the speed, for streaks.
- **Size Range:** Each instance's size, between Min and Max at random.
- **Over each instance's life:** Multiplies the size over each instance's own life (the default shrinks to nothing).
- **Shadows (Cast / Receive):** Whether the instances cast and receive shadows. Off is cheaper with many instances.

Performance

- **Info:** The inspector shows how many instances can be drawn, the batch size, and whether updates are Burst-compiled.
- **Jobs and Burst:** Instances are worked out in a job, Burst-compiled when the Burst package is installed. In the editor, 20,000 instances update in about 0.3 ms on Unity 6.0, with no garbage.

Shared Concepts Explained

Object Types List

Where you see "uses object types list", you can choose how the target object/position/direction is determined:

1. **User Defined:** Drag any GameObject from your scene or hierarchy.
2. **Main Camera:** Uses the transform of the Camera tagged MainCamera.
3. **This Object:** Uses the transform of the GameObject this FxChain component is attached to.
4. **Chain Parent:** Uses the previous component in the chain. (Not available if the current component is the Chain Root).
5. **From Chain Data:**Uses an object reference passed down via an external call or by another component higher in the chain (see the Starting Chains from Scripts (Manual Triggers) for additional info).
6. **X, Y, or Z axis:** Uses the world's positive X, Y, or Z direction vector.
7. **Own Trajectory:** (Available in specific contexts like Orientation) Uses the current movement direction of this GameObject.

Easing Types List

Where you see "uses easing types list", you can choose how the animation interpolates:

1. **Custom Easing Curve:** Define the interpolation using a standard Unity Curve Editor.
2. **Elastic:** Creates a springy, overshooting effect.
3. **Back:** Overshoots the target slightly before settling.
4. **Bounce:** Simulates bouncing off the target position before settling.

Chain Root Functionality: Trigger Modes

The Trigger Mode (Only available to the Chain Root) determines when and how the animation sequence begins:

- **AtStart:** Start the chain animation automatically when the scene is loaded (Play mode).
- **Manual:** Animation to be triggered via script, event or button etc.
- **Proximity:** Animation starts when within range of a specified object. Configure distance and target in the Timings section.
- **Custom Condition:** Animation triggers when a custom function returns true (see Custom Condition Triggers section).
- **Variable Watch:** Animation triggers when a monitored variable meets specified conditions (see Variable Watch System section).

Note on Custom Condition & Variable Watch Options:

FxChain includes an advanced trigger system that allows animations to respond dynamically to game state through custom conditions and variable watching. These triggers are configured programmatically and provide flexible, condition-based animation control.

To use these trigger modes, set the Chain Root's Trigger Mode to either Custom Condition or Variable Watch, then configure them via script as shown below.

Starting Chains from Scripts (Manual Triggers)

When using the Manual trigger mode, you can start an FxChain sequence from your own C# scripts by getting a reference to the root FxChain component and calling its initiation method. When doing so, you can optionally pass in data using the ChainData class to alter the initial pause and duration of the Chain Root or specify a material or objects to be used by chain links. This allows for dynamic, context-aware animation playback.

Call without data - *From a UnityEvent, Button etc*

```
using FxChain_Namespace;
public class CallFxChainSequenceToStart : MonoBehaviour
{
    public FxChain_Base chainRootScript;

    void Start()
```

```
    {
        chainRootScript.FxChain_CALL_ANIMATION_TO_START_noData();
    }
}
```

Call with data - *From a Script using the ChainData Class*

```
using FxChain_Namespace;
public class CallFxChainSequenceToStart : MonoBehaviour
{
    public FxChain_Base chainRootScript;
    public float initialPause = 1f;
    public float duration = 0f;
    public Material material;
    public GameObject targetObj;
    public GameObject orientObj_facing;
    public GameObject orientObj_up;
    public Vector3 animate_pos = Vector3.zero;
    public Vector3 animate_posRand = Vector3.zero;
    public Vector3 animate_rot = Vector3.zero;
    public Vector3 animate_rotRand = Vector3.zero;
    public Vector3 animate_scl = Vector3.zero;
    public Vector3 animate_sclRand = Vector3.zero;
    public int repeater_birthCount = 0;

    void Start()
    {
        chainRootScript.FxChain_CALL_ANIMATION_TO_START(new ChainData()
        {
            InitialPause = new Vector2(initialPause, initialPause),
            Duration = new Vector2(duration, duration),
            Material = material,
            TargetObj = targetObj,
            OrientObj_Facing = orientObj_facing,
            OrientObj_Up = orientObj_up,
            Animate_Pos = animate_pos,
            Animate_PosRand = animate_posRand,
            Animate_Rot = animate_rot,
            Animate_RotRand = animate_rotRand,
            Animate_Scl = animate_scl,
            Animate_SclRand = animate_sclRand,
            Repeater_BirthCount = repeater_birthCount,
        });
    }
}
```

Notes:

- Make sure to include the FxChain_Namespace when interacting with FxChain scripts.

- **Material:** Can be accessed by the SpriteSheetControl scripts. Select the Use Material From Chain option.
- **Target OBJ:**
  - Can be accessed by the Targeting section of Animate scripts. Select the From Chain Data option.
  - Can be used by Repeater as a reference object for the Birth Position. Select the From Chain Data option.
- **Orient OBJ (Facing and Up):** Can be accessed by the Orientation sections of Animate and Repeater scripts. Select the From Chain Data option.
- **ProximityObj:** Used in the Timing Controls section. Enable Proximity Override and select the From Chain Data option.
- **Animate Pos / Rot / Scl + Random Equivalents:** Allows you pass dynamic values to the Set Destination Transform section of an Animate component. Within the Animate component enable the “Use Data” toggle for the values you want to set this way.
- **Repeater_BirthCount:** Allows you pass a dynamic value to control how many spawns will be created. For example if it's a damage animation, you could pass in the number of times the character has been hit.
- **BranchPick:** Tells a Branch in From Chain Data mode which of its children to play (see Components: Branch).

Playback from Scripts (v3.0)

In 3.0 any component of a chain can control the whole chain from your own scripts: the calls act on its chain root. Set the chain root's Trigger Mode to Manual when a script plays it (with AtStart it also plays by itself). The 2.0 methods above (FxChain_CALL_ANIMATION_TO_START, FxChain_RESET_ANIMATION...) still work.

```
using FxChain_Namespace;
public class PlayMyEffect : MonoBehaviour
{
    public FxChain_Base effect;
    public GameObject target;

    async void Start()
    {
        //---Play the chain
        effect.Play();
        //---Play with per-play data
        effect.Play(new ChainData { TargetObj = target });
        //---Play, and wait for it to finish
        await effect.PlayAsync();
        //---Pause, resume and stop
        effect.Pause();
        effect.Resume();
        effect.Stop();
        //---Jump to 1.5 seconds into the effect
        effect.Seek(1.5f);
        //---Keep playing while the game is paused
        effect.TimeMode = FxChainTimeMode.Unscaled;
        //---Slow motion for this chain
        effect.Speed = 0.5f;
        //---Run code when the chain completes
        effect.ChainCompleted += chain => Debug.Log("Done");
    }
```

```
}
```

Playback Methods

- **Play() / Play(ChainData):** Plays the chain, optionally with per-play data (see Call with data above). To restart a chain that's playing, call Stop() first.
- **PlayAsync():** Plays the chain and returns an Awaitable that finishes when it completes, so a script can wait for the effect. If the chain is stopped first, the await throws an OperationCanceledException.
- **Pause() / Resume():** Holds the chain at the current time, then carries on from there.
- **Stop():** Stops the chain and resets it.
- **Seek(seconds):** Jumps to a time in the effect. It plays the chain on in fixed steps, so spawns, trails and events match a normal play.
- **AdvanceTime(seconds):** With the Manual time mode, moves the chain on by this much.

State and Time

- **IsPlaying, IsPaused, IsComplete:** The chain's state.
- **ElapsedTime:** Seconds since the chain's first frame. It follows the time mode and speed, and stops while paused.
- **Duration, NormalizedTime:** The planned length of one play, and ElapsedTime / Duration.
- **TimeMode:** Scaled follows Time.timeScale. Unscaled keeps playing when the game is paused (Time.timeScale 0), for pause menus; it follows Time.captureDeltaTime when that's set, so Unity Recorder captures play at the recorded frame rate. Manual moves only with AdvanceTime.
- **Speed:** A speed multiplier for this chain (0.5 = slow motion).

Events

- **C# Events:** ChainStarted, ChainCompleted, ChainRestarted, ChainStopped, ComponentLooped and ComponentCompleted.
- **FxChain_Events Component:** Add it to the chain root for the same events as UnityEvents, with no code.
- **Example Script:** FxC_Example_HelloFxChain shows the basics in a few lines.

Timeline Track

- **Setup:** With the Timeline package installed, add an FxChain Timeline Track, bind it to a chain and add clips.
- **Playback:** The chain follows the Timeline's playhead, in the editor and in Play mode.
- **Leaving a Clip:** Leaving a clip early resets the chain. Set a clip's post-extrapolation to Hold to keep its last state.

Custom Condition Trigger

Set a custom function that returns bool to control when animations trigger. This is ideal for complex logic involving multiple variables or game state combinations.

Method: **FxChain_SET_CUSTOM_CONDITION(**FxChain_TriggerCondition condition, MonoBehaviour sourceScript, string description**)**

Parameters:

- **condition -** Lambda function that returns bool (required).
- **sourceScript -** Reference to the calling script, typically “this” (optional but recommended).
- **description -** Human-readable description shown in Inspector (optional).

Best for:

- Complex logic combining multiple variables
- Infrequent condition checks
- Game state combinations (e.g., health + resources + abilities)

Example - *Simple Health Check*

```
using FxChain_Namespace;
public class SetupHealthWarningTrigger: MonoBehaviour
{
    public FxChain_Base healthWarningEffect;
    public int playerHealth = 100;
    public int healthWarningThreshold = 30;

    void Start()
    {
        // Trigger when health drops below 25
        healthWarningEffect.FxChain_SET_CUSTOM_CONDITION(() =>
            //---Condition logic (simple threshold check)
            {
                //---Compare current health against warning threshold
                return playerHealth < healthWarningThreshold;
            },
            //---Source script reference
            this,

            //---Condition description
            $"Health < {healthWarningThreshold}"
        );
    }
}
```

Example - *Complex Multi-Condition*

```
using FxChain_Namespace;
public class SetupPowerUpTrigger: MonoBehaviour
{
    public FxChain_Base powerUpEffect;
    public float mana = 100f;
    public bool hasPowerUp = false;
    public int powerLevel = 0;
    public float powerLevelRequired = 75f;

    void Start()
    {
        powerUpEffect.FxChain_SET_CUSTOM_CONDITION(() =>
            //---Condition logic (multiple condition check)
            {
```

```
                //---Combine two conditions:
                //---1. Power-up item is active (boolean state)
                //---2. Power level meets minimum requirement (threshold)
                return hasPowerUp && powerLevel >= powerLevelRequired;
            },
            //---Source script reference
            this,

            //---Condition description
            $"Has PowerUp AND Power >= {powerLevelRequired}"
        );
    }
}
```

Variable Watch Trigger

Continuously monitor a variable and trigger when it meets specified conditions using comparison operators. Includes configurable polling rates for performance optimization.

Method: **FxChain_WATCH_VARIABLE(**Func<float> valueGetter, MonoBehaviour sourceScript, string variableName = "")

Parameters:

- **valueGetter -** Lambda function that returns the value to monitor (required),
- **sourceScript -** Reference to the calling script, typically “this” (optional but recommended),
- **variableName -** Display name shown in Inspector (optional),

Comparison Operators:

- LessThan - Trigger when value **<** target
- LessThanOrEqual - Trigger when value **<=** target
- Equal - Trigger when value **≈** target
- NotEqual - Trigger when value **≠** target
- GreaterThanOrEqual - Trigger when value **>=** target
- GreaterThan - Trigger when value **>** target

Best for:

- Monitoring single values
- Threshold-based triggers (score milestones, temperature warnings)
- Configurable polling rates for performance control

Example - *Score Milestone*

```
using FxChain_Namespace;
public class SetupScoreMilestoneTrigger: MonoBehaviour
{
    public FxChain_Base scoreMilestoneEffect;
```

```
    public int currentScore = 0;
    public int scoreMilestone = 1000;

    void Start()
    {
        scoreMilestoneEffect.FxChain_WATCH_VARIABLE(() =>
                //---Value getter (simple score monitoring)
                currentScore,

                //---Source script reference
                this,

                //---Display name (shown in editor)
                "Player Score"
            )
            .WithCondition(
                //---Comparison type (triggers when reaching or exceeding milestone)
                ComparisonOperator.GreaterThanOrEqual,

                //---Threshold value (1000 points required)
                scoreMilestone
            )
            //---Polling Rate: 5Hz (score changes aren't frequent)
            .PollingRate(5f);
    }
}
```

Example - *Temperature Monitoring*

```
using FxChain_Namespace;
public class SetupTemperatureAlertTrigger: MonoBehaviour
{
    public FxChain_Base temperatureAlertEffect;
    public float engineTemperature = 20f;
    public float maxTemperature = 80f;

    void Start()
    {
        temperatureAlertEffect.FxChain_WATCH_VARIABLE(() =>
                //---Value getter (continuous temperature monitoring)
                engineTemperature,

                //---Source script reference
                this,

                //---Display name (includes unit)
                "Engine Temperature (°C)"
            )
```

```
                .WithCondition(
                    //---Comparison type (triggers on overheat)
                    ComparisonOperator.GreaterThan,

                    //---Threshold value (80°C maximum)
                    maxTemperature
                )
                //---Polling Rate: 20Hz (critical system monitoring)
                .PollingRate(20f);
        }
    }
```

Polling Rates (Performance Optimization)

The polling rate determines how often the variable watch system checks conditions (in Hz - times per second). Choose appropriate rates based on your needs:

- **1-3 Hz:** Background checks, non-critical conditions
- **5-10 Hz:** Standard gameplay triggers (recommended for most use cases)
- **15-30 Hz:** Critical gameplay conditions requiring responsive checks
- **30+ Hz:** Very responsive checks (use sparingly for performance)

Advanced Example - *Multiple Conditions Combined*

```
using FxChain_Namespace;
public class SetupUltimateAbilityTrigger: MonoBehaviour
{
    public FxChain_Base ultimateAbilityEffect;
    public int playerLevel = 1;
    public float mana = 100f;
    public float maxMana = 100f;
    public bool isGrounded = true;
    public List<string> unlockedAbilities = new();
    public int enemiesNearby = 0;


    void Start()
    {
        ultimateAbilityEffect.FxChain_SET_CUSTOM_CONDITION(() =>
            //---Condition logic (ultimate ability requirements)
            {
                bool hasEnoughMana = mana >= maxMana * 0.8f;
                //---Resource check (80% mana minimum)
                bool hasAbility = unlockedAbilities.Contains("Ultimate");
                //---Ability unlock status (must be unlocked)
                bool highLevel = playerLevel >= 10;
                //---Character progression (level 10 minimum)
                bool enemiesPresent = enemiesNearby >= 3;
                //---Combat situation (3+ enemies required)
```

```
                bool isGroundedAndStable = isGrounded;
                //---Player state (must be on ground)

                //---All conditions required for ultimate
                return hasEnoughMana && hasAbility && highLevel &&
                    enemiesPresent && isGroundedAndStable;
            },
            //---Source script reference
            this,

            //---Condition description (all requirements)
            $"Ultimate Ready: Level 10+, 80% Mana, Unlocked, 3+ Enemies, Grounded"
        );
    }
}
```

Advanced Example - *Chain of Conditions*

```
using FxChain_Namespace;
public class SetupPhasedEffectsTriggers : MonoBehaviour
{
    public FxChain_Base phase1Effect;
    public FxChain_Base phase2Effect;
    public FxChain_Base phase3Effect;
    public int currentPhase = 1;
    public int playerLevel = 1;
    public float experience = 0f;

    void Start()
    {
        //---Phase 1: Experience threshold
        if (currentPhase == 1)
        {
            phase1Effect.FxChain_WATCH_VARIABLE(() =>
                //---Value getter (basic XP monitoring)
                experience,

                //---Source script reference
                this,

                //---Display name (indicates phase)
                "Phase 1 - Experience Points"
            )
            .WithCondition(
                //---Comparison type (triggers on XP milestone)
                ComparisonOperator.GreaterThanOrEqual,

                //---Threshold value (initial milestone)
```

```
                    100f
                )
                //---Polling Rate: 5Hz (standard XP monitoring)
                .PollingRate(5f);
        }
        //---Phase 2: Experience + phase check
        if (currentPhase == 2)
        {
            phase2Effect.FxChain_SET_CUSTOM_CONDITION(() =>
                //---Condition logic (phase 2 requirements)
                {
                    //---Check current progression phase
                    bool correctPhase = currentPhase == 2;     //---Must be exactly phase 2

                    //---Check increased XP requirement
                    bool enoughXP = experience >= 250f;        //---250 XP threshold

                    //---Require both phase and XP conditions
                    return correctPhase && enoughXP;
                },
                //---Source script reference
                this,

                //---Condition description (shows both requirements)
                $"Current Phase: 2 AND Experience >= 250"
            );
        }
        //---Phase 3: Level requirement
        if (currentPhase == 3)
        {
            phase3Effect.FxChain_SET_CUSTOM_CONDITION(() =>
                //---Condition logic (final phase requirements)
                {
                    //---Check for final phase
                    bool correctPhase = currentPhase == 3;     //---Must be in final phase

                    //---Check level requirement
                    bool highLevel = playerLevel >= 20;          //---Level 20 minimum

                    //---Require both final phase and level
                    return correctPhase && highLevel;
                },
                //---Source script reference
                this,

                //---Condition description (shows final requirements)
                $"Final Phase (3) AND Level >= 20"
            );
        }
    }
}
```

Editor Visualization

When using Custom Condition or Variable Watch trigger modes, the Chain Root inspector displays:

- **Custom Condition Mode:**
  - Status indicator showing if a condition function is assigned
  - Instructions for setup
- **Variable Watch Mode:**
  - Variable name
  - Comparison operator
  - Target value
  - Polling rate (Hz)
  - **During Play Mode:** Live current value and condition status (TRUE/FALSE)

Example Scripts

FxChain includes three comprehensive example scripts demonstrating the trigger system:

- **FxC_Example_TriggerModes01_WithData.cs -** Demonstrates how to pass custom data when manually triggering animations via scripts.
- **FxC_Example_TriggerModes02_CustomConditions.cs** - Four practical examples including health warnings, power-up checks, combo multipliers, and physics-based conditions.
- **FxC_Example_TriggerModes03_VariableWatch.cs** - Seven variable watching patterns including score milestones, temperature monitoring, speed triggers, distance calculations, performance ratings, multi-threshold health checks, and equality conditions.
- **FxC_Example_TriggerModes04_AdvancedTriggers.cs** - Advanced patterns including dynamic trigger switching, ultimate ability requirements, performance optimizations, achievement systems, and phased trigger sequences.

Each example script includes comprehensive inline documentation and editor test helpers for quick experimentation.

Performance Best Practices

1. **Choose Appropriate Polling Rates:** Don't poll every frame if not needed - lower rates save CPU cycles.
2. **Cache Expensive Calculations:** For complex computations, cache results and update periodically rather than calculating every check.
3. **Use Custom Conditions for Complex Logic:** When combining many variables, custom conditions can be more efficient as you control exactly when evaluation happens.
4. **Disable Triggers When Not Needed:** Set the FxChain component's enabled property to false to stop Update() from running when triggers aren't needed.

Chain Root Functionality: Other Settings

Destroy Chain Objects when Animation Completes

Enabled: Once all chain components have completed their animations destroy all relevant game objects (runtime only).

Note: FxChain features a staggered object destruction system. This is to prevent frame dips when destroying many objects during a single frame. You can adjust the number of objects destroyed each frame with Destroy Objects Per Frame (below).

Playback Frame Rate

These values control how often the system runs through the update loop and can improve performance.

- **Min:** If the runtime frame rate falls below this value, playback is clamped to the minimum so motion eases instead of hitching.
- **Max:** Caps how fast the animation evaluates each frame; lower values cut CPU cost, higher values keep timing crisp.

Time and Speed

- **Time:** Scaled follows Time.timeScale. Unscaled keeps playing when the game is paused (Time.timeScale 0), for pause menus. Manual moves only when a script calls AdvanceTime.
- **Speed:** A speed multiplier for the whole chain (0.5 = half speed).

Restart When Complete

Plays the whole chain again each time it finishes. It isn't available while Destroy Chain Objects When Complete is on.

Destroy Objects Per Frame

How many finished objects are destroyed per frame in Play mode (default 10). In 2.0 this was Destroy Object Batch Size in the Other section.

Chain Doctor Summary

When the Chain Doctor finds a problem in the chain, a summary with a red Open button shows at the top of Chain Root Options (see Editor Tools).

Upgrade Chain to v3.0 Behaviour

Shown when the chain has components made with 2.0. They keep v2.0 behaviour so the effect looks exactly as before, and the button moves the whole chain to v3.0 behaviour (Undo is available). See Upgrading from v2.0.

Chain Root Functionality: Preloading Spawns

Instantiating many GameObjects simultaneously (especially with complex hierarchies) via Repeaters can cause performance spikes. The root FxChain component has an option to Preload Spawns. This finds all Repeaters in the chain and prepares their instances ahead of time (e.g., during a loading screen). Activate the chain normally afterwards for smooth playback. This isn’t always needed but it’s there as an option if you do.

Preloading isn't available with Recycle Spawns. For effects that play often, Recycle Spawns (on for new Repeaters) is usually the better choice: finished spawns are kept and reused.

Chain Root Functionality: Creating Prefabs Workflow

The root FxChain component includes a Create Prefab button. This handles the specific considerations needed to correctly package an entire FxChain hierarchy into a Unity Prefab, ensuring all internal references within the chain are maintained. Use this button instead of manually dragging the Chain Root into the Project window.

- **Process:**
  1. Click the Create Prefab button (available only on the Chain Root) to open a dedicated window.
  2. Specify the save location and name for your prefab asset.
  3. The tool gathers all GameObjects belonging to the FxChain sequence starting from the root, including any helper objects created by components (like Ignore Parent Orientation nulls or Repeater spawn containers).
  4. It temporarily creates a new parent GameObject in the scene and arranges the collected chain objects under it.
  5. This temporary hierarchy is saved as a prefab asset using PrefabUtility.SaveAsPrefabAsset, preserving the internal FxChain links.
  6. The original chain objects and the temporary parent are removed from the scene, and an instance of the newly created prefab is placed in the scene.
- **External Parents:** If any objects in your chain have parent GameObjects that are *not* part of the FxChain sequence itself, the tool will warn you. These external parents will not be included in the prefab, which might affect the prefab's behavior if it relied on transforms inherited from those parents.
- **Preloading:** If you have used the Enable Preloading option, the *structure* required for preloading is saved in the prefab, but the preloaded spawn instances themselves are not. Preloading will still need to be initiated at runtime or via the button on the prefab instance.
- **Usage:** The resulting asset is a standard Unity prefab and can be instantiated into your scenes as needed.
- **In Place (v3.0):** The chain becomes the prefab instance where it is (Undo restores it); 2.0 rebuilt it at the scene origin.

Editor Tools

Windows and helpers for building, checking and previewing chains.

Welcome Window

- **Opens:** Once per project after FxChain is imported or updated, and any time from Tools → FxChain → Welcome.
- **Contents:** The Getting Started guide, a button per demo scene, the optional packages (installed only if you choose to), what's new, the documentation, the FxChain Discord and buttons that add the AI guide to Claude or copy it for other assistants.

Getting Started Guide

Tools → FxChain → Getting Started, or the button at the top of the Welcome window. A guide for new users that builds a gem pickup effect in eleven short steps, all in the Inspector.

- **The Steps:** Add an Animate, make the gem rise and spin, slow it to 2 seconds, preview it, add a ring of sparkles (Repeater, Arc), give each spawn a shape, make the sparkles float away (Random -/+), stagger them (Spawn Timing), add a trail, flash a light, and see it all in the Chain Timeline.
- **Ticks Itself Off:** Each step is checked against the scene, so steps can be done in any order. A step is done once it has every value it asks for: its card flashes and the next step opens.
- **Show Me:** Selects what the step is about.
- **Do It For Me:** Does the step for you (Undo works as usual).
- **Practice Scene:** It starts in a new, unsaved scene so your own scenes aren't changed, or in the open scene.

Chain Timeline

Window → FxChain → Chain Timeline, or the Timeline button next to Play Animation In Editor.

- **Bars:** The selected chain as a tree of bars, one per component, coloured by type: when each starts, how long it runs and its loops.
- **Spawns:** Lighter parts show start times spread across spawns, outlined bars are Repeater spawn windows, and ticks mark loops.
- **Scrub:** Drag on the ruler or a bar to scrub the preview to any moment. The preview matches what plays at runtime.
- **Toolbar:** Play, Pause and Stop; a search field that filters the components; Chain Doctor; and Lock, which keeps the window on this chain.
- **Play Mode:** It shows and seeks the running chain.

Scene View Playback Overlay

- **Shown:** In the Scene view while an FxChain object is selected.
- **Controls:** Play, Pause, Stop, a Timeline button and a scrub slider with the current time.
- **Chain Doctor Button:** When the Chain Doctor finds a problem in the chain, a ! button appears beside Timeline: red for errors, yellow for warnings only. Click it to open the Chain Doctor.

Chain Doctor

Window → FxChain → Chain Doctor. Lists problems in the open scenes' chains, with Select and Fix buttons:

- **Errors:** For example a Repeater with nothing to spawn, a Mass Repeater without a mesh or material, a proximity trigger without a target, or a Splines arrangement without the Splines package.
- **Warnings:** For example a component whose duration is 0, a material property that isn't on the shader, or a mesh without Read/Write for builds.
- **Info:** For example components still using v2.0 behaviour (with an Upgrade button), or a recycling Repeater that spawns your own scripts.
- **Show Info / Refresh:** Shows or hides the info notes, and checks again.
- **Summaries:** A chain root shows a summary at the top of its Chain Root Options, and the Scene view overlay shows the ! button.

Chain Templates

- **Gallery:** Window → FxChain → Chain Templates shows prefabs holding FxChain chains, with previews.
- **Add:** Click one to add it at the Scene view's focus point.
- **Save Selected Chain:** Saves the selected chain as a template.
- **Folder:** Templates live in FxChain/Templates; the Folder button picks another.

Scene View Handles

- **Animate:** The end point (drag it to set the movement; the box shows the random range), which way it faces, and its shake range.
- **Animate Follow Path:** The From and To points, dragged along the path.
- **Proximity:** The proximity trigger distance.
- **Repeater:** Spawn locations as preview objects or gizmos.
- **Look At, Physics Impulse, Camera Shake:** The target, the push or blast radius, and how far the shake reaches.

Inspectors

- **UI Toolkit:** Inspectors are built with UI Toolkit, as Unity's own are, and fields move onto the next line in a narrow inspector.
- **Sections:** Sections remember whether they're open. Blue titles sit over each group of settings, with an on / off toggle beside the optional ones.

- **Timing Bar:** Every component's Timing section shows where it plays in the whole chain (see Timings Section - Timing Bar).
- **TIPS:** Each section has TIPS at the bottom describing every option.
- **Play Mode Status:** In Play mode, the chain root shows the chain's time, time mode and speed.
- **Multi-Object Editing:** Select several FxChain objects to edit them together.
- **Classic (IMGUI) Inspectors:** Preferences → FxChain switches back to the 2.0 drawing and names.

Renamed Options

The same settings, with clearer names in the 3.0 inspectors (2.0 name → 3.0 name):

- **Timings Controls, Spawn Timings Controls** → Timing, Spawn Timing.
- **Set Destination Transform, Switch to Follow Spline Mode** → Transform: Destination / Follow Path.
- **Preview End Transform / End at Initial Transform** → Start at Destination.
- **Pos / Rot / Scl Random, Scale Separate Axes** → Random -/+, Scale Per Axis.
- **Use World Space: Use World Movement / Rotation** → World Space: Movement / Rotation.
- **Transform Easing, Target Easing, Blending Between Transforms** → Easing, Approach Easing, Blend with Destination.
- **Targeted Orientation, Facing Obj/Direction, Up obj/direction** → Aim / Orientation, Face Towards, Up Direction.
- **Object with Spline Component, Orient Along Spline, Enable Banking, Using Animated Spline Positions** → Path, Face Along Path, Banking, Animated Path.
- **Birth Count and Origin, Spawn Selection Behaviour, Birth Transform Object Reference** → Spawns, Child Selection, Birth Origin.
- **Set Arrangement Type, Progressive / Random Transforms, Placement Reference Objects** → Arrangement, Progressive / Random Offsets, Spawn Preview.
- **SpriteSheet List, Set Duration from SpriteSheets, PlaySpeed FPS, Set Last Frame to Null** → Sprite Sheets, Duration From Sprite Sheets, Frames Per Second, Clear Sprite at End.
- **External Script Trigger** → On Animation Start.

Preferences

Edit → Preferences → FxChain (the FxChain Editor Preferences button in the Other section opens it):

- **Limit Decimal Places:** Decimal places shown in FxChain number fields.
- **Scale Slider Ranges:** Multiplies the range of FxChain's sliders.
- **Hide TIPS:** Hides the TIPS foldouts in FxChain inspectors.
- **Use Global Section Toggles:** On: a section opened or closed in one FxChain inspector is opened or closed in all of them. Off: each component remembers its own.
- **Classic (IMGUI) Inspectors:** Draws FxChain's inspectors the way 2.0 did.
- **Hide Movement Gizmo when Previewing:** Hides the animation path gizmo during editor previews.
- **Auto-Reset Preview When Complete:** Resets an editor preview a moment after it finishes. Off: it holds the last frame until you click in the Scene view, press a key or press Reset.
- **Rename Includes Script Name:** FxChain names objects "[position]_Script_Name"; off leaves out the script name.
- **Verbose Logging:** Logs FxChain's internal steps to the Console, for bug reports.

Search, Shortcuts and Stats

- **Unity Search:** Type fxc: in the Search window to list the FxChain components in the open scenes.
- **FxChain: Play or Stop:** Alt+Shift+P plays or stops the selected chain's preview (rebind it in Edit → Shortcuts).
- **Context Menu:** A component's menu has Select Chain Root, Play or Stop and Open Chain Timeline.
- **FxChain_StatsHUD:** Add it to a scene for an in-game readout of FxChain's cost. In the Profiler, FxChain's work shows as FxChain.ChainTick, FxChain.CreateSpawn, FxChain.Setup and FxChain.DestroyBatch.
- **Presets:** Unity Presets work with FxChain components. When saving a preset of a component that's part of a chain, exclude its Next Scripts In Chain list, or applying it will link the other component to this chain's children.

AI Assistant

FxChain/Documentation/FxChain_AI_Guide.md teaches Claude, ChatGPT and other AI assistants how FxChain works: the chain and timing model, every component, the scripting API, recipes and common pitfalls. Tools → FxChain → AI Assistant puts it where your assistant reads it (the Welcome window has the same buttons):

- **Add to Claude Code (This Project):** Copies the guide to .claude/skills/fxchain/SKILL.md in your project (next to Assets, not imported by Unity). Claude Code then loads it whenever FxChain comes up; commit the folder to share it with your team.
- **Add to Claude Code (All Projects):** Copies it to .claude/skills/fxchain/SKILL.md in your home folder, so Claude Code knows FxChain in every project.
- **Make a Claude App Skill:** Saves fxchain.zip where you choose and offers to open Claude's settings. Upload it under Settings → Capabilities → Skills (Features in some versions) to use the guide in claude.ai and the desktop and mobile apps. Skills need a paid Claude plan with code execution turned on.
- **Copy the AI Guide:** Puts the guide's text on the clipboard, to paste into ChatGPT, Gemini, Copilot or any other assistant, or into a project's knowledge or instructions.
- **Show the AI Guide File:** Selects the guide in the Project window.
- **IDE Assistants:** In Cursor, Copilot and similar, add the file as a rule or context file (for example .cursor/rules/fxchain.md).

Demo Scenes and Templates

Eleven genre scenes with 44 effects, each also a ready-made template.

Scenes 08 to 18 in FxChain/ExampleScenes each show four effects ("stations") that games in one genre need. Every station's chain is also a prefab in FxChain/Templates/<Genre>: open Chain Templates and click one to add it to your scene, then swap in your own meshes and materials. The scenes need URP; stations with text need TextMeshPro's essentials (Window → TextMeshPro → Import TMP Essential Resources). Scenes 01 to 07 are the 2.0 demos.

The Scenes

- **08 Arcade & Platformer:** Coin Block, Jump Pad, Slime Hop, Checkpoint.
- **09 Melee & Action:** Sword Combo, Ground Slam, Critical Hit, Dash Afterimages.
- **10 Shooter & Explosions:** Bullet Impacts, Explosive Barrel, Muzzle Burst, Bullet-Hell Spiral.
- **11 Fantasy & RPG:** Elemental Bolt, Power-Up Aura, Summoning Portal, Loot Chest.
- **12 UI & Menus:** Menu Open / Close, Reward Popup, Notification Toast, Pause Menu.
- **13 Puzzle, Cards & Casual:** Match-3 Clear, Card Deal & Flip, Prize Wheel, Pack Opening.
- **14 Strategy & Tower Defense:** Construction, Unit Recruit, AoE Telegraph, Chain Lightning.
- **15 Adventure, Survival & Horror:** Haunted Corridor, Campfire, Locked Door & Key, Chop & Collect.
- **16 Sci-Fi & Tech:** Teleporter, Hologram, Shield Hit, Turret Lock-On.
- **17 Life Sim & Farming:** Tree Growth, Crop Field, Watering, Market Sale.
- **18 Racing, Sports & Party:** Boost Pad, Race Countdown, Goal Celebration, Dice & Board Move.

The Panel

- **Stations:** A button per station (or keys 1-9); Space replays.
- **Scrub and Speed:** A scrub slider (it calls Seek) and a speed slider.
- **Slow Motion:** S sets Time.timeScale to 0.25 (chains on Unscaled time ignore it).
- **Hide:** H hides the panel. Left alone, the scene plays the stations in turn.
- **Features:** Each station lists the components and features it's built with.
- **Scripts:** The panel is FxC_DemoController in Scripts/ExampleScripts. Stations that need code use small station player scripts there too (dice, prize wheel, gold counter, pause, repeats).

Recipes

- **Pop Text In and Out:** An Animate whose scale curve goes 0, overshoot, 1, 0, with a Text child. Set the Animate's scale easing to Custom Easing Curve: by default rotation and scale follow the position curve.
- **Collect and Fly to a Counter:** Repeater spawns that hop, then an Animate target blend pulls them to the counter (Coin Block, Market Sale).
- **A Wave Across a Board:** A Grid Repeater with a Distance Based initial pause measured from a corner or centre object (Match-3 Clear, Crop Field).
- **A Result Chosen by Game Code:** Play(new ChainData(BranchPick: i)) with a Branch in From Chain Data mode (Bullet Impacts, Elemental Bolt, Dice).
- **Land on a Chosen Result:** ChainData(Animate_Rot: ...) read by an Animate with Use Data on its rotation (Prize Wheel, the die's face).
- **Random or Rare Outcomes:** Branch Random or Weighted (Goal Celebration, Critical Hit, Loot Chest).
- **Combo Strings:** Branch Sequence plays the next child each time (Sword Combo).
- **React to the Player:** Children that wait for a Proximity target (Haunted Corridor).
- **Play When a Value Reaches a Threshold:** Variable Watch (Unit Recruit plays when gold reaches the cost).
- **Menus While the Game Is Paused:** Unscaled time, and await PlayAsync() before resuming (Pause Menu).
- **Ambience That Keeps Going:** Loop Forever on each part (Campfire).
- **Hide Something Until Its Moment:** Give it a zero start scale, or switch it with an Event (Explosive Barrel).

Troubleshooting

Example Scene Materials are Pink / Shadows are Missing

If you notice that materials in the FxChain example scenes appear solid pink, or if shadows seem to be missing or incorrect, it typically indicates a mismatch between the project's Render Pipeline settings and the Universal Render Pipeline (URP) materials used by the examples. FxChain is designed and tested using URP. To Resolve:

In 3.0 the demo scenes are made for URP only. FxChain itself works with the Built-in Render Pipeline, URP and HDRP, and the materials it makes for itself (a new Trail's line, Material Properties' renderer) use the render pipeline in use.

Verify Unity Version:

Ensure you are using Unity 6.0 LTS or a newer version. FxChain relies on features and APIs available in these versions.

Check Your Project's Render Pipeline:

1. FxChain requires the Universal Render Pipeline (URP) to be active in your project.
2. Go to Edit → Project Settings → Graphics.
3. Look at the Scriptable Render Pipeline Settings field.
   - If this field is empty or says "None": Your project is likely using the Built-in Render Pipeline. You will need to either install and configure URP in your existing project (refer to Unity's official URP documentation for the migration process) or test FxChain in a dedicated URP project.
   - If this field has an asset assigned: Click the asset. Check if it's a UniversalRenderPipelineAsset. If it is, URP is correctly configured. If it's a different type (e.g., HDRenderPipelineAsset), you are using a different Scriptable Render Pipeline, which is incompatible with the FxChain example materials.

Confirm with a Clean URP Project (Recommended Test):

The most reliable way to confirm that FxChain itself is working correctly and that the issue stems from project settings is to test it in a fresh URP project.

1. Open the Unity Hub:
2. Click the blue New project button.
3. Select your desired Editor Version (ensure it's 6.0 LTS or newer).
4. Choose the 3D (URP) template (the exact name might vary slightly, e.g., "Universal Project Template"). Do not select the "3D (Core)" or "Built-in" template.
5. Create the project.
6. Import the FxChain asset into this new project.
7. Open the example scenes and verify if the materials and shadows now render correctly.

If the examples work correctly in the new URP project, the issue lies within the configuration of your original project's render pipeline. If the problem persists even in a clean URP project, there might be a different issue, and you should consider reaching out for further support.

Compilation Errors After Uninstalling the Unity Splines Package

Symptom Error Message:

The type or namespace name 'Splines' does not exist in the namespace 'UnityEngine' (are you missing an assembly reference?)

In 3.0: FxChain no longer uses a Scripting Define Symbol for Splines (see Additional Info below), so this error shouldn't appear. If you upgraded from 2.0, you can remove the old UNITY_SPLINES symbol it added as follows.

Alternatively:

1. Go to Edit → Project Settings.
2. Select the Player category from the list on the left.
3. In the Inspector window, find the Other Settings section and expand it if necessary.
4. Scroll down to the Scripting Define Symbols field.
   a. Important: Ensure you are editing the symbols for the correct Build Target platform (e.g., Standalone, Android, iOS). You might need to check and remove the symbol from multiple platforms if you switch targets often.
5. Locate and remove UNITY_SPLINES.
   a. Make sure to also delete any redundant semicolon (;) next to it.
   b. **Example:** Change SYMBOL_A;UNITY_SPLINES;SYMBOL_B to SYMBOL_A;SYMBOL_B.
6. Click the Apply button at the bottom of the Inspector. Unity will recompile. Check the Console window again; the errors related to the uninstalled package should now be resolved.

Additional Info:

FxChain can optionally integrate with the Unity Splines package. To manage this, relevant code is wrapped within a custom symbol like this:

```
#if FXCHAIN_SPLINES

    // Code specific to the Unity Splines package

#endif
```

In 3.0 the symbol is FXCHAIN_SPLINES, and FxChain's assembly definitions set it themselves while the Splines package is installed, so nothing is added to your Project Settings.

When the package is uninstalled, Unity removes the symbol with it. FxChain 2.0 added a UNITY_SPLINES symbol to the Project Settings instead, which could be left behind and cause the error above; it can be removed safely as described.

The Scene View is Covered in FxC Icons (This shouldn’t be an issue anymore)

These can be hidden using the Scene view's Gizmos menu:

1. Click the Gizmos button in the Scene view toolbar (top right).
2. Find the FxChain script icon (e.g., for FxChain_Base or similar) in the list.
3. Click the icon image next to the script name to toggle its visibility off.

Stuttering During Playback when a Repeater Generates Spawns

This can be avoided by preloading spawns from the Chain Root.

Or turn on Recycle Spawns on the Repeater (on for new Repeaters), so finished spawns are reused instead of created again.

A Mass Repeater Draws Nothing

- **Mesh and Material:** Both must be set (Chain Doctor flags this).
- **Lifetime:** Must be above 0.
- **GPU Instancing:** Without it on the material, a copy with it on is drawn. Turn it on with the inspector's button to avoid the copy.

Mesh Arrangements Spawn Nothing in Builds

Mesh Points, Mesh Surface and the Mass Repeater's Mesh Surface shape read the mesh's vertices, which player builds only keep when the mesh's Read/Write import setting is enabled. Chain Doctor warns about meshes without it.

Text, UI Animate, Volume Weight, Decal or VFX Control Does Nothing

These need their Unity packages: TextMeshPro / Unity UI (Text, UI Animate), URP or HDRP (Volume Weight), URP with the Decal renderer feature (Decal) and Visual Effect Graph (VFX Control). Without them they compile and do nothing, and the inspector says what's missing.

Camera Shake or Physics Impulse Doesn't Preview

Both play in Play mode only: editor previews don't move the Scene view camera, and physics doesn't run in editor previews.

Play Animation In Editor Does Nothing

Check the timing bar in the chain root's Timing section: a chain whose components all have a Duration of 0 finishes at once. Chain Doctor lists components with a duration of 0.

Get in Touch

- Having issues and troubleshooting isn’t helping?
- Found a bug?

- Have feedback? Good? Bad? Hopefully good.
- Have a feature or functionality that you'd love to see added to FxChain?
- Want to share something cool you've made with the tool?

Discord Community: **https://discord.gg/6NuvwfBPrq**

Email: You can reach me at: **jamie** [AT] **incendiary** [DOT] **design**

Reddit: **incendiary-design**

Unity Asset Store: **https://assetstore.unity.com/packages/slug/316031**

Made Something Cool with FxChain?

Send me a DM through Instagram and I'll share it on the channel - #madewithfxchain