Energy VFX · Procedural Energy Beams & Flows System

• • •

Support & Feedback

Email: care@skulmovski.studio • Discord: [Join our community]

If you encounter any issues, please contact me via Discord or Email before submitting a review.

• • •

Table of Contents

About Energy Flow..... 2
⚠️ IMPORTANT: First Time Setup (Read Before Starting)..... 4
Project Requirements..... 4
Using Setup Assistant..... 5
Fast Integration (Quick Start / Demo Scene)..... 7
Setting Up Energy Flow in Your Own Scene..... 8
Endpoints — Point, Line & Surface..... 9
Using Surface Endpoints..... 11
Understanding the Energy Flow Controller..... 13
Spawn & Transfer Behavior..... 15
Customizing the Flow — PRE, Motions & POST..... 18
Appearance & Child Effects..... 21
UFlow Presets..... 25
Runtime Control / C# API..... 28
Production Optimization..... 33
Known Behavior, Limitations & Troubleshooting..... 33
Technical Reference: Parameter Glossary..... 36
Thank You!..... 43

About Energy Flow

Procedural Energy Beams & Flows VFX System is a configurable Visual Effect Graph system for creating particle-based energy transfer effects between user-defined **Start** and **End** locations.

Instead of providing only a collection of fixed effects, the asset is built around a reusable flow system. You define where the effect begins, where it travels, and how the particles behave along the path.

Start and End positions can use three different endpoint types:

- **Point** — a single Transform position.
- **Line** — a position sampled along a line defined by two Transforms.
- **Surface** — positions sampled across the surface of a Mesh or Skinned Mesh.

Start and End types are configured independently, allowing setups such as **Point → Point**, **Line → Point**, **Surface → Point**, **Surface → Surface**, and other combinations. The current endpoint data exposes Point, Line and Surface as the three supported position modes.

Flow Layers

The system contains three independently configurable Flow Layers.

Each layer has its own:

- Spawn settings
- Start and End endpoints
- Transfer behavior
- PRE path modifiers
- Procedural Motions
- POST path modifiers
- Appearance settings
- **Child particle effect**

The controller processes the same configuration structure independently for each active layer, allowing a simple effect to use only one layer or a more detailed effect to combine several different flows.

For example, one layer can form the main energy stream while another adds spiral strands, turbulence, lightning-like motion, or other secondary detail.

Procedural Path Customization

The path between Start and End can be shaped using several procedural motion types, including:

- Bend
- Oscillation
- Spiral
- Lightning
- Organic Flow
- Ribbon Weave

- Branch Split
- Turbulence

These motions can be combined rather than being limited to a single motion type. PRE and POST modifiers provide additional control over how the resulting motion develops along the transfer path.

The supplied motion data contains all eight motion groups as part of each Flow Layer.

Built for Customization

The included prefabs and Demo Scene examples represent only a small selection of what can be created with the system.

Each Flow Layer exposes approximately **135 configurable settings**, covering not only particle appearance, but also spawning, endpoint behavior, transfer, path shaping, procedural motion, variation, and secondary effects.

Because these settings can be combined across three independent Flow Layers, the supplied examples should be treated as **starting points and references rather than fixed effect templates**.

You can modify an existing example or build a completely different effect using the same underlying system.

UFlow Presets

Custom Energy Flow configurations can be saved as reusable **UFlow presets**.

Presets can be stored locally inside the project or exported in a portable format together with their required texture resources, making it possible to move effects between projects or share custom configurations with other users.

Secondary Effects

Each Flow Layer also includes its own **Child Effect** for adding secondary particles to the main flow.

Child particles can be triggered:

- when a parent particle spawns;
- at a selected point during its transfer;
- as it approaches the destination;
- continuously while it travels.

This can be used for effects such as launch particles, sparks along the path, intermediate bursts, or impacts at the target.



⚠️ IMPORTANT: First Time Setup (Read Before Starting)

Before opening the Demo Scene or setting up Energy Flow in your own scene, we **highly recommend running the included VFX Setup Assistant first**.

The Assistant performs the basic checks required for the asset to work correctly and provides quick access to the main setup tools.

If the Assistant does not open automatically after importing the asset, open it manually from:

Tools > Skulmovski Studio > Energy VFX · Beams And Flows · [URP or HDRP] > VFX Setup Assistant

Important: Use the Energy Flow version that matches your project's active Render Pipeline. URP and HDRP versions are separate and should not be mixed in the same project.

Project Requirements

Energy Flow requires:

- **Visual Effect Graph** — must be installed through the Unity Package Manager.
- **Supported Render Pipeline** — use the version of the asset that matches your project.

The main Asset Store package is the standalone **URP version**.

Render Pipeline Compatibility

Energy Flow is supplied in two separate pipeline-specific versions:

- **URP** — the main Asset Store package and a complete standalone version for Universal Render Pipeline projects.
- **HDRP** — a separate standalone `.unitypackage` included inside the main URP package for High Definition Render Pipeline projects.

Use only the version that matches the active Render Pipeline of your Unity project.

The URP version is intended only for **URP projects**, and the HDRP version is intended only for **HDRP projects**.

Do not use or open the URP version in an HDRP project, and do not use or open the HDRP version in a URP project. The pipeline-specific VFX Graphs, shaders, and materials are not interchangeable, and using the wrong version can prevent the effects from working correctly or produce shader and VFX Graph errors.

After importing the HDRP package into an HDRP project, it works as a standalone version and does not require the URP version to remain in that project. For an HDRP project, import the included HDRP `.unitypackage` and use that version instead of the URP content.

VFX Graph Variants

Energy Flow includes several pipeline-specific VFX Graph variants. All variants use the same Energy Flow Controller and configuration workflow; the main difference is how the particles are rendered.

URP

The URP version includes two VFX Graph variants:

- **Unlit** — uses unlit particle rendering and does not react to scene lighting.
- **Lit** — uses lit particle rendering and reacts to scene lighting.

HDRP

The HDRP version includes three VFX Graph variants:

- **Unlit** — uses unlit particle rendering and does not react to scene lighting.
- **Lit · Standard** — uses HDRP's **Standard** material type. This provides the more complete lighting model and is the most performance-intensive of the supplied HDRP variants.
- **Lit · Simple Lit** — uses HDRP's **Simple Lit** material type, providing a lighter lit alternative when the full Standard material model is not required.

Choose the variant that best matches the visual requirements and performance budget of your project.

Using Setup Assistant

Before manually setting up Energy Flow in your project, we highly recommend running the built-in **VFX Setup Assistant**.

The Assistant performs the essential project and scene checks required by the asset, provides direct actions where configuration can be completed automatically, and gives quick access to the tools commonly used while setting up Energy Flow.

After making changes outside the Assistant, click **Refresh** to update the checklist and verify the current project or scene state.

Project-Wide Settings

These checks apply to the Unity project rather than the currently open scene.

The Assistant verifies that:

- the active Render Pipeline matches the imported Energy Flow package;
- the **Visual Effect Graph** package is installed and available.

Visual Effect Graph is required to simulate and render the supplied effects. If the package is missing, the Assistant provides direct access to the Unity Package Manager so it can be installed before continuing.

It is recommended to resolve all required **Project-Wide Settings** immediately after importing Energy Flow and before opening the Demo Scene or beginning a custom setup.

Scene Settings

Scene Settings evaluate the currently open scene.

The Assistant checks whether the supplied default **Energy Flow** prefab is already present. If it is missing, the **Add Energy Flow** action can be used to add the default setup directly to the active scene.

The default prefab is the recommended starting point for a custom effect. It already contains the required Energy Flow components and provides a working configuration that can then be adapted to your own Start and End objects.

Because Scene Settings always refer to the **currently open scene**, run the Assistant again after switching from the Demo Scene to your own scene.

Useful Tools

The Assistant also provides quick access to the tools most commonly needed while working with the asset:

- **Recompile VFX Graphs** — forces the supplied VFX Graph assets to reimport. This can be useful if a graph does not compile correctly immediately after the package is imported.
- **Explore the Demo** — opens the included Demo Scene so you can inspect the supplied examples and configurations.
- **Surface Sample Baker** — opens the editor tool used to prepare Mesh and Skinned Mesh surfaces for **Surface** endpoints.

These tools are optional during normal Point or Line workflows, but remain available from the Assistant when required.

Additional Notes

The **Additional Notes** section contains short reminders related to the normal Energy Flow workflow, Surface endpoints, presets, and other asset-specific behavior.

The Setup Assistant is intended to cover the **essential first-time project and scene setup**. Detailed configuration of Start and End endpoints, Surface baking, Flow Layers, motion settings, and other effect controls is covered in the corresponding sections of this documentation.

Fast Integration (Quick Start / Demo Scene)

The included Demo Scene is the fastest way to confirm that Energy Flow has been imported correctly and to explore the supplied effect examples before creating your own setup.

For the first launch, we recommend following this sequence:

Step 1: Complete the Project-Wide Setup

Open the **VFX Setup Assistant** and resolve the required **Project-Wide Settings** first.

Step 2: Open the Demo Scene

Once the project-wide checks are complete, use **Explore the Demo** in the Setup Assistant to open the included Demo Scene.

The effects are previewed directly in the Editor, so entering Play Mode is not required.

Step 3: Explore the Example Categories

In the Demo Scene Hierarchy, find the **Examples** object.

It contains several child categories with preconfigured Energy Flow examples, such as:

- **Moving**
- **Impact**
- **Sweep**
- **MoreExamples**
- **Motions**
- **Complex**
- **CharAnim**
- **Vertical**

The **Motions** category is enabled by default.

Enable the category you want to inspect and browse the prefabs inside it to see how different Energy Flow configurations are built.

The examples demonstrate different combinations of endpoints, spawning behavior, procedural motions, Flow Layers, Child Effects, and other controller settings.

Step 4: Inspect the Example Settings

Select any example prefab in the Scene and inspect its **Energy Flow Controller** settings.

The Demo Scene is intended not only as a visual showcase, but also as a practical reference for building your own effects. You can compare different examples and see how their controller settings produce different types of motion and particle behavior.

Once you are familiar with the supplied examples, continue with the next section to set up Energy Flow in your own scene.

Setting Up Energy Flow in Your Own Scene

After the project-wide checks are complete, you can add Energy Flow to any custom scene.

The recommended starting point is the supplied default prefab, since it already contains the required **Visual Effect** and **Energy Flow Controller** components.

Step 1: Add Energy Flow to the Scene

Open the **VFX Setup Assistant** in the scene where you want to use the effect.

Under **Scene Settings**, use **Add Energy Flow** if the default Energy Flow prefab is not already present.

The prefab can then be positioned anywhere in the scene. Its own Transform does not define the actual Start and End of the flow — those are assigned separately through the controller.

Step 2: Choose a Flow Layer

Select the Energy Flow object and open the **Energy Flow Controller**.

The controller contains three independently configurable Flow Layers:

- **Detail Flow A**
- **Detail Flow B**
- **Detail Flow C**

For a basic setup, you only need one enabled Flow Layer. The additional layers can be used later to add more visual detail or combine different flow behaviors.

Step 3: Assign Start and End

Open the **Endpoints** section of the Flow Layer you want to configure.

Start and End are configured independently, and each can use one of three modes:

- **Point**
- **Line**
- **Surface**

For the simplest setup, select **Point** for both Start and End.

Assign one Transform to **Start Point** and another Transform to **End Point**. The effect will use their world positions as the beginning and destination of the flow.

The assigned objects can be moved freely. Energy Flow updates its source data continuously, so moving endpoint Transforms can be used directly for effects between moving objects.

Step 4: Preview the Result

Energy Flow uses an **ExecuteAlways** controller, allowing the effect and its endpoint references to update directly in the Editor. Entering Play Mode is not required for normal visual setup and preview.

Move the assigned Start or End objects and adjust the Flow Layer settings while viewing the result in the Scene or Game view.

Once the basic Point-to-Point setup is working, you can replace either endpoint with a **Line** or **Surface** source depending on the effect you want to create.

The next section explains the three endpoint types and how each one is configured.

Endpoints — Point, Line & Surface

Each Flow Layer has its own independent **Start** and **End** endpoint configuration.

Start and End can each use one of three modes:

- **Point**
- **Line**
- **Surface**

The Start and End modes are selected independently, so different endpoint types can be combined within the same Flow Layer.

Point

Point mode uses a single **Transform** as the endpoint position.

Assign the required Transform to the corresponding **Point** field.

Point endpoints are useful for effects between specific positions, such as:

- object to object;
- hand to target;
- weapon to enemy;
- portal center to character;
- any other single Start or End position.

Moving the assigned Transform automatically moves the endpoint.

Line

Line mode defines the endpoint using two Transforms:

- **Line Start**
- **Line End**

Particles can use positions distributed along the line between these two Transforms.

Both Transforms can be moved independently, allowing the line to change position, direction, and length while the effect is active.

Line endpoints are useful when the flow should originate from or arrive across an elongated region rather than a single point.

Surface

Surface mode allows particles to use positions distributed across a Mesh or Skinned Mesh.

Unlike Point and Line endpoints, a Surface must first be prepared using the included **Surface Sample Baker** and assigned through a **Flow Surface Source**.

The complete Surface workflow is covered in the next section.

Mixing Endpoint Types

Start and End do not need to use the same mode.

For example:

- **Point → Point**
- **Line → Point**
- **Point → Line**
- **Surface → Point**
- **Point → Surface**
- **Surface → Surface**

Choose the combination that best matches how the effect should connect to the objects in your scene.

Endpoint Tracking

Endpoint type and endpoint tracking are separate settings.

Frozen and **Live** tracking determine how already-created particles react when their Start or End endpoint moves.

Tracking behavior is covered later in **Spawn & Transfer Behavior**.

Using Surface Endpoints

Surface mode allows Energy Flow particles to use positions across the surface of a Mesh or Skinned Mesh instead of a single Transform or Line.

Before an object can be used as a Surface endpoint, its geometry must be prepared with the included **Surface Sample Baker**. The resulting Surface Sample Asset is then used by a **Flow Surface Source** component at runtime.

Surface endpoints can be used independently for Start and End, allowing setups such as **Surface → Point**, **Point → Surface**, or **Surface → Surface**.

Step 1: Open the Surface Sample Baker

Open the Baker from the **VFX Setup Assistant** under **Useful Tools**, or manually from:

Tools > Skulmovski Studio > Energy VFX · Beams And Flows · [URP or HDRP] > Surface Sample Baker

Step 2: Select the Root Object

Drag the object you want to use as a Surface into the **Root** field.

The Baker searches the selected Root and its children for supported renderers:

- **Mesh Renderer**
- **Skinned Mesh Renderer**

If the object is made from several separate meshes, they can be baked together into a single Surface Sample Asset.

For animated characters, select the character root that contains the required Skinned Mesh Renderers and their skeleton hierarchy.

Step 3: Choose the Renderers

The **Renderers** list shows the supported meshes found below the selected Root.

Enable the renderers that should participate in the Energy Flow surface and disable any parts that should be ignored.

For example, a character may contain separate renderers for the body, clothes, hair, weapons, or accessories. You can choose only the parts that should be available for particle spawning or targeting.

Use **All**, **None**, or **Refresh** to quickly manage the renderer selection.

Step 4: Choose the Sample Count

Sample Count controls how many sampling points are distributed across the selected surface.

The samples are distributed according to triangle surface area, so larger parts of the selected geometry naturally receive more samples.

A larger Sample Count provides a denser set of available surface positions. A smaller value may be sufficient for simple objects or effects that use relatively few particles.

The default value is **10,000 samples** and the Baker supports values up to **100,000**.

There is no single correct value for every mesh. Use enough samples to produce the surface coverage required by your effect.

Step 5: Bake the Surface

Click **Bake Selected Surface**.

Choose where the generated Surface Sample Asset should be saved inside the project.

The asset stores the data required by Energy Flow to reconstruct the selected surface and sample it at runtime. The selected renderers are combined into a single internal proxy representation for the Surface Source.

Step 6: Add a Flow Surface Source

Add a **Flow Surface Source** component to the same Root object that was used when baking the surface.

Assign the generated **Surface Sample Asset** to the component.

For Skinned Mesh surfaces, the baked asset stores bone paths relative to the selected Root. The Flow Surface Source uses those paths to follow the character's skeletal animation at runtime.

Once initialized, the Flow Surface Source continuously resolves the current surface data used by Energy Flow.

Step 7: Assign the Surface to Energy Flow

In the **Energy Flow Controller**, open the required Flow Layer and its **Endpoints** section.

Set the required Start or End **Mode** to **Surface**, then assign the prepared **Flow Surface Source** to the corresponding Surface field.

The same Surface Source can be used as either the Start or End of the flow depending on the required effect.

⚠️ Important Notes

The original source meshes must have **Read/Write** enabled for the Surface Sample Baker to access their geometry. If Read/Write is disabled on one of the selected meshes, the Baker cannot process that renderer.

Normal skeletal deformation from a **Skinned Mesh Renderer** is supported. However, **Blend Shape deformation is not currently copied into the generated proxy**, so **Blend Shape changes are not represented by Surface sampling**.

If the source geometry, renderer selection, mesh topology, or skinning setup is changed, bake a new Surface Sample Asset so the stored surface data matches the updated object.

Understanding the Energy Flow Controller

The **Energy Flow Controller** is the main component used to configure the effect.

It controls playback, presets, endpoint references, and all visual and behavioral settings for the three independent Flow Layers.

The controller is divided into several main sections.

UFlow Preset

The preset section is used to load, save, reset, and export complete Energy Flow configurations.

Applying a UFlow preset changes the effect settings while keeping the current scene-specific **Point, Line, and Surface endpoint references** intact. This makes it possible to try different visual presets without reconnecting the effect to your objects each time.

The complete UFlow workflow is covered later in this documentation.

Playback

The **Playback** section controls how Energy Flow starts and how long it continues emitting particles during runtime.

Play On Enable

Play On Enable determines whether the Energy Flow Controller automatically starts the effect when it becomes enabled in **Play Mode**.

- **Enabled** — the effect starts automatically when the controller becomes enabled.
- **Disabled** — the controller keeps the effect stopped until it is started manually through the runtime API with `Play()` or `Play(float duration)`.

⚠️ **Play On Enable affects runtime playback only.**

In **Edit Mode**, the supplied Energy Flow VFX Graphs can still preview automatically through their Visual Effect **Initial Event (`OnPlay`)**. This editor preview is independent from the **Play On Enable** setting.

Therefore, an effect may be visible and animated while editing the scene even when **Play On Enable** is disabled.

Playback Mode

Two runtime playback modes are available:

- **Infinite** — the effect continues emitting particles until emission is stopped manually or the controller is disabled.
- **Timed** — the effect emits particles for the specified **Duration**, then stops creating new particles automatically.

Stopping emission does not immediately remove particles that are already alive. Existing particles can continue their movement until their individual lifetime is complete.

Duration

Duration defines how long the controller emits particles when **Playback Mode** is set to **Timed**.

For example:

- `Mode = Infinite` → Duration is not used to stop emission.
- `Mode = Timed`, `Duration = 3` → new particles are emitted for 3 seconds, after which emission stops.
- `Play On Enable = Off` → neither Infinite nor Timed playback begins automatically; call `Play()` manually when the effect should start.

Starting the Effect Manually

When **Play On Enable** is disabled, the effect can be started from code:

`Play()` starts the effect using the **Mode** and **Duration** configured in the Inspector.

`Play(float duration)` starts a **Timed** playback using the supplied duration, regardless of the Inspector playback mode.

Editor Preview

The controller also provides preview controls for working with the effect without entering Play Mode:

- **Play / Retrigger** — starts or restarts the editor preview.
- **Stop Emission** — stops generating new particles while allowing already spawned particles to finish.
- **Clear Preview** — stops and clears the current preview.

Flow Layers

The controller contains three independent Flow Layers: **Layer A**, **Layer B**, and **Layer C**.

All three layers use the same set of settings and can be enabled, disabled, and configured independently.

Each Flow Layer contains:

- **Spawn** — controls how particles are generated.
- **Endpoints** — defines the Start and End sources.
- **Transfer** — controls how particles travel between Start and End.

- **PRE** — modifies how procedural motion is evaluated along the transfer path.
- **Motions** — contains the procedural motion types used to shape the path.
- **POST** — applies additional shaping after the procedural motion.
- **Appearance** — controls the visual appearance of the particles.
- **(!) Child** — controls the secondary particle effect generated from the parent flow.

A simple effect may use only one Flow Layer. Additional layers can be enabled when different particle styles, motion patterns, timing, or secondary details need to be combined into a more complex effect.

The following sections explain the individual Flow Layer systems and their settings in more detail.

Spawn & Transfer Behavior

Each Flow Layer has its own **Spawn** and **Transfer** settings.

Spawn controls **when and how many particles are created**, while Transfer controls **how those particles move from the selected Start endpoint to the End endpoint**.

Spawn Modes

Four Spawn Modes are available:

Mode	Behavior
Variable	Continuously emits particles while periodically choosing a new emission rate from the configured Rate Range.
Periodic Burst	Repeatedly emits particle bursts using the configured Count Range and Delay Range.
Single Burst	Emits a single burst using the configured particle count and delay range.
Constant	Continuously emits particles at a fixed rate.

Each Flow Layer selects its Spawn Mode independently, so different layers of the same effect can use completely different emission behavior.

Sweep

Sweep delays the beginning of the transfer across the **Start endpoint**.

Available modes are:

- **Instant**
- **Top Down**
- **Bottom Up**
- **Left Right**
- **Right Left**

Sweep does not affect **Point** endpoints. When the Start Mode is **Point**, all Sweep modes behave as Instant.

For a **Line** endpoint, Sweep is evaluated using the line's position along the **world X and Y axes**.

If the line has no extent along the selected Sweep axis, the particles receive the same Sweep delay instead of a directional progression across the line.

For a **Surface** endpoint, Sweep uses the normalized coordinates stored when the surface is baked. The directions therefore follow the baked Surface Root's local X and Y orientation.

Sweep Duration determines how long it takes for the transfer to progress across the source.

Sweep adds a per-particle delay before transfer begins rather than changing particle spawning itself.

Start Delay & Pre Fade

Start Delay adds a waiting period before a particle begins travelling toward the End endpoint.

Pre Fade Duration controls how the particle appears during that waiting period.

Unlike its name may suggest, **Pre Fade Duration is not measured in seconds**. It represents a normalized portion of the particle's transfer delay.

If there is no transfer delay, the particle becomes visible immediately.

Start & End Tracking

Start and End tracking are configured independently using **Frozen** or **Live**.

Start	End	General Behavior
Frozen	**Frozen**	Each particle keeps the Start and End positions captured when it was created.
Frozen	**Live**	The particle keeps its original Start position while continuing to react to movement of the target.
Live	**Frozen**	The Start side can react to movement of the source while the target remains fixed to its captured position.

Live	**Live**	Both sides of the transfer can react to movement of their assigned endpoints.

Frozen is useful when particles should preserve the position associated with their own spawn moment.

Live is useful when an already existing flow should continue responding to a moving source or target.

Transfer Speed & Particle Lifetime

Speed controls how quickly particles travel from Start to End.

Energy Flow does not use one fixed travel lifetime for every particle. The initial travel duration is calculated automatically from the resolved distance between Start and End:

Travel Duration = Distance / Speed

Because of this, moving the target farther away increases the time required for the particle to reach it, while a closer target results in a shorter travel duration.

The particle lifetime also includes its transfer delay, so a particle waiting because of Start Delay or Sweep remains alive long enough to complete the waiting period before travelling.

This behavior is especially important when using effects that temporarily move away from the source before following the main path, such as **Spawn Kick**.

Path Follow

Path Follow controls how strongly the procedural Motion offset contributes to the path followed by the particles.

Intermediate values reduce the influence of the procedural shape.

The additional **Loose Time**, **Arrival Start**, and **Final Time** controls are primarily used with **Spawn Kick** when Follow is enabled. They control how quickly a kicked particle is corrected back toward its calculated path and how that correction changes as it approaches the destination.

Launch

The **Launch** section is available for effects that use a **Surface** as the Start endpoint.

None uses the normal transfer behavior.

Spawn Kick gives particles an initial velocity away from the Start Surface before the normal transfer takes over.

The kick direction follows the sampled **surface normal**, allowing particles to peel away from a character or object before travelling toward the End endpoint.

- **Kick Speed** controls the strength of the initial velocity along the surface normal.

- **Inherit Surface Velocity** adds movement from the animated or moving Start Surface to the particle's initial velocity.
- **Kick Follow Enabled** determines whether the particle is corrected back toward the calculated transfer path after the initial kick.

Launch / Spawn Kick is only used when `Start Mode = Surface`. It has no effect when the Start endpoint is Point or Line.

Customizing the Flow — PRE, Motions & POST

The shape of an Energy Flow path is built in three stages: **PRE → Motions → POST**

PRE modifies the data used to generate the procedural path.

Motions create the actual bends, waves, spirals, noise, and other offsets from the direct Start → End trajectory.

POST modifies the final combined shape after all active Motions have been added together.

This structure makes it possible to build substantially different effects without changing the underlying transfer system.

Combining Motions

The Motion types are **additive**.

You are not limited to selecting one Motion at a time. Multiple Motions can be configured together, and their offsets are combined into a single procedural path before POST processing is applied.

For example, a flow can combine a broad **Bend**, a small **Spiral**, and animated **Turbulence** to create a path that would be difficult to reproduce with any one Motion alone.

The system currently provides eight Motion types:

Bend

Creates up to two controlled bends along the transfer path.

Each bend has its own position, amount, width, and angle, allowing anything from a simple arc to more complex S-shaped paths.

Oscillation

Creates wave-like movement around the path using two independent oscillation components.

Different amplitudes, frequencies, phases, and axis rotation can be combined to produce simple waves, elliptical movement, or more complex oscillating patterns.

Spiral

Wraps the path around the direct transfer line.

Radius controls how far the spiral moves away from the base path, while **Turns** controls how many rotations occur between Start and End. The spiral can also rotate over time.

Lightning

Creates a segmented, irregular path suitable for electrical or unstable energy effects.

Amount controls the strength of the displacement, **Segments** controls the segmentation of the shape, and **Refresh Rate** can periodically generate a new variation of the lightning path.

Organic Flow

Adds smooth, noise-based deformation to the path.

It is useful for less mechanical movement such as magical energy, smoke-like flows, souls, or other naturally moving effects.

Ribbon Weave

Distributes particles between several rotating strands around the central path.

Strand Count defines the available strand positions; it does **not** create additional particles by itself. Existing particles are distributed between those strands.

Branch Split

Separates particles into multiple procedural branches.

Branch Count defines the available branch directions rather than multiplying the particle count. Spread, twisting, and branch shaping can then be adjusted independently.

Turbulence

Adds animated multi-scale noise to the trajectory.

It is useful for breaking up otherwise smooth shapes or adding smaller irregular movement on top of Bend, Spiral, Ribbon, and other Motions.

PRE

PRE settings modify the values used by the Motion system **before the Motion offsets are calculated**.

Because PRE is shared by the layer, the same controls can influence several active Motions at once.

Progress

Progress Warp remaps the normalized progress along the path.

Progress Steps converts smooth progress into discrete steps, allowing procedural shapes to change in a more segmented or stylized way.

Window

The Window controls where along the Start → End path the Motion system is allowed to have an effect.

Window Start and **Window End** define the active region, while **Window Softness** controls how gradually the influence enters and leaves that region.

Window Peak Position shifts where the strongest part of the shared Motion envelope occurs.

Modulation

Frequency Over Progress changes the spatial frequency of supported Motions as they travel along the path.

Phase Over Progress shifts the phase or angular progression of Motions that use phase-based calculations.

Variation

Strength introduces per-particle variation into the procedural path.

Animation

Time Scale controls the shared time used by animated Motions.

Strobe Rate quantizes that Motion time, replacing completely smooth animation with discrete updates when a value above zero is used.

POST

POST is evaluated **after all active Motion offsets have been combined**.

Shape

Amount Over Progress controls the overall strength of the combined Motion shape along the path.

Radius Over Progress changes the radial scale of the combined shape along the path.

Axis Scale independently scales the two axes perpendicular to the transfer direction. This can turn circular procedural shapes into flatter or more elliptical forms.

Source Expand

Source Expand controls how the procedural offset develops immediately after the Start.

With higher **Expand Strength**, the path can begin tightly around the source and gradually open into its full procedural shape by **Expand End**.

Target Collapse

Target Collapse does the opposite near the End.

Collapse Start defines where convergence begins, while **Collapse Strength** controls how strongly the procedural offset is reduced toward the target.

At full strength, the Motion offset approaches zero at the end of the transfer, allowing wide or chaotic paths to converge cleanly on the target.

Target Collapse changes the **shape of the path**; it is not a separate target-tracking or arrival system.

Pulse

Pulse periodically changes the strength of the combined path deformation.

It can create repeated widening and narrowing, breathing motion, or pressure-wave-like variation along the flow.

The effect can be controlled through **Strength**, **Frequency**, **Phase**, and animation **Speed**.

Roll

Roll rotates the entire combined Motion shape around the direction of the transfer path.

Appearance & Child Effects

After the path and movement are configured, the **Appearance** and **Child** sections control how each Flow Layer is rendered and what secondary particles it can generate.

Each of the three Flow Layers has its own independent Appearance and Child settings.

Appearance

Appearance controls the visual presentation of the main particles without changing their transfer path.

Texture & Size

Texture defines the particle texture used by the layer.

Size Range randomizes the initial size between the minimum and maximum values.

Size Over Life then modifies that size throughout the particle's lifetime.

Color & Alpha

Color And Alpha Over Life controls both particle color and transparency throughout the lifetime.

Color Intensity multiplies the resulting color intensity and can be used to make energy particles brighter or more subdued.

Orientation

Two orientation modes are available:

- **Billboard** — particles face the camera.
- **Along Velocity** — particles align with their direction of movement.

Angle Z Range adds randomized Z rotation to the main particles.

Material Settings

Smoothness and **Metallic** control the corresponding material properties when using the Lit particle output.

Alpha Clipping removes pixels below the selected alpha threshold. Increasing it produces harder edges and removes more semi-transparent parts of the texture.

Fade Distance softens intersections between particles and scene geometry, helping avoid harsh clipping where particles pass through surfaces.

Child Effects

Every Flow Layer also contains its own **Child** particle effect.

Child particles are generated from particles of their parent Flow Layer and can be used to add secondary details such as:

- sparks emitted along the flow;
- particles released from the source;
- intermediate bursts;
- impact particles near the target;
- continuously emitted secondary trails or fragments.

The Child system is divided into **Trigger**, **Motion**, and **Appearance** settings.

Child Trigger

The Trigger determines **when** Child particles are created.

Off

Disables Child particle generation for the selected Flow Layer.

On Spawn

Generates a Child burst when the parent particle is created.

This is useful for secondary particles that should appear directly at the beginning of the flow.

On Progress

Generates a Child burst when the parent particle crosses the selected **Progress** value.

This can be used to create secondary effects at a specific part of the path.

On Arrival

Generates a Child burst as the parent particle approaches the end of its transfer.

The trigger occurs near the end of the path rather than waiting for the particle simulation to fully complete.

This is useful for impacts, sparks, flashes, or other reactions near the target.

Over Time

Continuously generates Child particles while the parent particle is actively travelling.

Rate controls the average emission rate.

Child particles are not generated by Over Time while the parent is waiting before transfer or after its transfer has finished.

Burst Count, Rate & Spawn Chance

Burst Count determines how many Child particles are generated by event-based triggers such as **On Spawn**, **On Progress**, and **On Arrival**.

Rate is used by **Over Time** instead.

Spawn Chance behaves slightly differently depending on the Trigger Mode.

For event-based triggers, Spawn Chance determines whether the entire Child burst occurs.

For **Over Time**, it reduces the average number of generated Child particles rather than simply enabling or disabling one complete burst.

This makes it possible to introduce controlled randomness without changing the parent Flow Layer itself.

Child Motion

After a Child particle is generated, its Motion settings control how it moves independently from its parent.

Direction Mode

Three direction modes are available:

- **Random**
- **Flow Direction**
- **Custom**

Random uses a direction selected from the configured Random Direction range.

Flow Direction launches the Child particle relative to the direction in which its parent Flow particle is travelling.

Custom uses the manually specified **Custom Direction** vector.

Reverse

Reverse flips the selected direction when using **Flow Direction** or **Custom**.

It does not affect the current Random direction mode.

Spread

Spread adds randomized sideways deviation around the selected **Flow Direction or Custom Direction**.

At `0`, particles retain the selected direction more closely. Increasing the value creates a wider dispersion.

Spread is not additionally applied when **Direction Mode = Random**, because Random already uses its own direction range.

Speed, Lifetime & Size

Speed Range randomizes the initial velocity of newly created Child particles.

Lifetime Range determines how long they remain alive.

Size Range determines their initial size.

Together these settings make the Child effect independent from the size, lifetime, and motion of its parent particles.

Gravity & Drag

Gravity applies a directional acceleration to Child particles.

It can be used for falling sparks, rising fragments, sideways forces, or other directional movement.

Drag gradually reduces particle velocity. Higher values cause Child particles to lose their initial momentum more quickly.

Child Appearance

Child particles have their own Appearance settings and do not need to visually match their parent Flow Layer.

They provide independent controls for:

- **Texture**
- **Size Over Life**
- **Color And Alpha Over Life**
- **Color Intensity**
- **Orientation Mode**
- **Smoothness**
- **Metallic**
- **Alpha Clipping**
- **Fade Distance**

This makes it possible, for example, to use large soft particles for the main flow while generating small sharp sparks from the same layer, or to use a completely different texture and color for an arrival effect.

UFlow Presets

Energy Flow configurations can be saved as reusable **UFlow presets** using the `.uflow` format.

A preset stores the visual and behavioral configuration of the controller, including **Playback**, all three Flow Layers, their Spawn, Transfer, PRE, Motions, POST, Appearance settings, and their Child effects.

UFlow presets are useful for creating a personal effect library, reusing configurations between objects, or transferring custom effects between Unity projects.

What a UFlow Preset Does Not Store

Endpoint setup is intentionally kept separate from the preset.

Applying a preset does **not** replace:

- Start or End Mode;
- Point references;
- Line references;
- Surface references.

This allows the same effect configuration to be applied to completely different objects without replacing their scene-specific endpoint setup.

For example, a preset created using one character can later be applied to another Energy Flow Controller while keeping that controller's own Start and End surfaces.

Applying a Preset

To use an existing preset:

1. Assign the `.uflow` asset to **UFlow Preset**.
2. Click **Apply**.

The stored Playback and Flow Layer settings are copied to the controller immediately.

Existing endpoint setup remains unchanged.

Save As..

Save As.. creates a project-local `.uflow` preset from the current controller configuration.

Use this when you want to save a custom effect for continued use inside the same Unity project.

Project-local presets reference their textures through the existing Unity project assets. The texture files themselves are **not duplicated**.

After saving, the newly created preset is automatically assigned to the controller.

Export Preset..

Use **Export Preset..** when the preset needs to be moved to another Unity project or shared with another user.

Portable export creates the `.uflow` file and copies the project textures required by the effect into a matching resource folder.

For example:

MyPreset.uflow

MyPreset_Res/

- required texture files
- corresponding `.meta` files

The `_Res` folder uses the same name as the `.uflow` file and must remain beside it.

If the preset does not require external texture resources, a resource folder may not be necessary.

Sharing a Portable Preset

When sharing an exported preset, include:

- the `.uflow` file;
- its matching `_Res` folder, if one was created;



- the included `.meta` files.

Do not rename or separate the `_Res` folder from its matching `.uflow` preset.

When moved into another Unity project's **Assets** folder, Unity automatically imports the `.uflow` file as an Energy Flow preset and resolves its exported texture resources from the matching `_Res` folder.

The preset can then be assigned to **UFlow Preset** and applied normally.

Save As vs Export Preset

Save As.. is intended for presets that stay inside the current project.

It keeps references to the existing project textures and avoids creating duplicate resource files.

Export Preset.. is intended for portable presets.

It creates self-contained texture resources so the effect does not depend on the original project's texture GUIDs.

Reset

Reset restores the controller settings from the supplied default Energy Flow preset.

As with normal preset application, the current endpoint setup is preserved.

This provides a quick way to return the controller to its default configuration without reconnecting Start and End objects.

Texture Note

UFlow presets should use regular **Texture2D assets stored in the project**.

Unity built-in texture references are not suitable for new portable presets because their original sub-assets cannot be identified reliably when transferred between projects.

If a built-in texture is used, replace it with a regular project texture before saving or exporting the preset.

Runtime Control / C# API

Energy Flow exposes a public runtime API for starting and stopping effects, changing controller settings, updating individual Flow Layers, refreshing endpoints, and reacting to playback events.

The main component is `EFlowController`, and the three runtime layers are addressed through `RuntimeLayer.DetailFlowA`, `DetailFlowB`, and `DetailFlowC`.

Namespaces

```
using SkulmovskiStudio.VFX_Energy.EnergyFlow.Scripts.EFlow;
using SkulmovskiStudio.VFX_Energy.EnergyFlow.Scripts.EFlow.Data;
```

Starting and Stopping the Effect

Play()

Starts the effect using the **Playback Mode** and **Duration** currently **configured in the controller**.

```
energyFlow.Play();
```

Before playback begins, the controller reapplies its current settings and refreshes endpoint data.

`Play()` is a runtime method and only starts the effect in **Play Mode**.

Play(float duration)

Starts a Timed playback using the supplied duration, regardless of the Playback Mode currently selected in the Inspector.

```
energyFlow.Play(2.5f);
```

This example emits particles for 2.5 seconds and then automatically stops emission. **Already spawned particles can remain alive until their simulation finishes**.

StopEmission()

Stops the creation of new particles without immediately clearing particles that are already alive.

```
energyFlow.StopEmission();
```

The controller then waits until the VFX Graph no longer has active systems before reporting the effect as completed.

Playback Properties

Playback can also be configured from code:




```
energyFlow.PlayOnEnable = false;
energyFlow.PlaybackMode = PlaybackMode.Timed;
energyFlow.PlaybackDuration = 3f;
```

The public playback properties are:

Property	Description
PlayOnEnable	Automatically starts the effect when the controller becomes enabled in Play Mode.
PlaybackMode	Selects Infinite or Timed playback.
PlaybackDuration	Duration used by Timed playback.
PlaybackData	Access to the complete Playback settings structure.

`PlaybackDuration` is clamped to zero or higher.

Playback State

Two public properties can be used to inspect the current runtime state:

IsEmitting is true while the controller is actively emitting new particles.

IsWaitingForCompletion becomes true after emission has stopped while previously spawned particles or Child effects are still alive.

For example:

```
if (energyFlow.IsEmitting)
{
    // The effect is currently emitting.
}
```

Playback Events

The controller exposes three runtime events:

Event	Called When
Started	Playback has been started or retriggered.
EmissionStopped	The controller stops creating new particles.

Completed	Emission has stopped and the VFX Graph no longer contains active systems.

Example:

```
private void OnEnable()
{
    energyFlow.Started += OnFlowStarted;
    energyFlow.Completed += OnFlowCompleted;
}

private void OnDisable()
{
    energyFlow.Started -= OnFlowStarted;
    energyFlow.Completed -= OnFlowCompleted;
}

private void OnFlowStarted()
{
    Debug.Log("Energy Flow started.");
}

private void OnFlowCompleted()
{
    Debug.Log("Energy Flow completed.");
}
```

Accessing Flow Layers

The complete settings for each layer are available through:

```
energyFlow.LayerA
energyFlow.LayerB
energyFlow.LayerC
```

Each property returns the corresponding `LayerData`, including Spawn, Endpoints, Transfer, PRE, Motions, POST, Appearance, and Child settings.

Some Energy Flow setting groups, such as `SpawnData`, are value types (`struct`). When modifying them through the runtime API, copy the settings, modify the copy, assign it back to the layer, and then call the corresponding Apply method.

For example, changing the Constant Spawn Rate of Detail Flow A:

```
var layer = energyFlow.LayerA;
var spawn = layer.spawn;

spawn.spawnMode = FlowSpawnMode.Constant;
spawn.constantRate = 50f;

layer.spawn = spawn;

energyFlow.ApplySpawnSettings(EFlowController.RuntimeLayer.DetailFlowA);
```

`LayerData` itself is a class, while settings such as `SpawnData` are structures stored inside it.

Applying Runtime Changes

Changing a value in `LayerData` does not by itself guarantee that the new value has already been sent to the VFX Graph.

The API therefore provides several Apply methods.

Method	Updates
`ApplySettings()`	All three Flow Layers, all Child settings, and source data.
`ApplyLayerSettings(layer)`	One complete Flow Layer and its Child effect.
`ApplyLayerEnabled(layer)`	Enabled state only.
`ApplySpawnSettings(layer)`	Spawn settings only.
`ApplyTransferSettings(layer)`	Transfer settings only.
`ApplyPreSettings(layer)`	PRE settings only.
`ApplyMotionSettings(layer)`	Motion settings only.
`ApplyPostSettings(layer)`	POST settings only.
`ApplyAppearanceSettings(layer)`	Main particle Appearance only.
`ApplyChildSettings(layer)`	Complete Child configuration.
`ApplyChildTriggerSettings(layer)`	Child Trigger only.
`ApplyChildMotionSettings(layer)`	Child Motion only.

`ApplyChildAppearanceSettings(layer)`	Child Appearance only.

Use **ApplySettings()** for occasional large changes.

When changing settings frequently at runtime, use the most specific Apply method required instead of reapplying the complete controller.

Changing Endpoints at Runtime

Endpoint references can be modified through the selected layer's `endpoints` object.

For example:

```
var endpoints = energyFlow.LayerA.endpoints;

endpoints.startMode = PositionMode.Point;
endpoints.startPoint = newStart;

endpoints.endMode = PositionMode.Point;
endpoints.endPoint = newEnd;

energyFlow.RefreshSources(EFlowController.RuntimeLayer.DetailFlowA);
```

`RefreshSources(layer)` immediately pushes the current endpoint configuration of the selected Flow Layer to the VFX Graph.

To refresh all three Flow Layers at once:

```
energyFlow.RefreshSources();
```

Point, Line, and Surface endpoint data is normally refreshed automatically by the controller during its update cycle, so `RefreshSources()` does not need to be called every frame.

Use it when endpoint modes or references are changed through code and the new configuration needs to take effect immediately.

Production Optimization

VFX Graph Bounds

The supplied Energy Flow VFX Graphs use **Automatic Bounds** by default.

Automatic Bounds are convenient while creating and configuring effects because the visible area can change significantly depending on the assigned Start and End endpoints, their movement, procedural Motions, and Child particles.

However, Automatic Bounds require Unity to update the effect bounds at runtime.

For a finalized production effect with a predictable movement area, consider replacing Automatic Bounds with **Recorded** or carefully configured **Manual Bounds**.

Do not use fixed bounds that are smaller than the actual runtime effect area. If Start or End can move outside the recorded region, or the distance between them can become significantly larger at runtime, the effect may be culled incorrectly.

For highly dynamic or unpredictable setups, keeping **Automatic Bounds** may be the safer option.

Known Behavior, Limitations & Troubleshooting

The Effect Does Not Appear After Import

Energy Flow is available in separate **URP** and **HDRP** versions.

Make sure the installed Energy Flow version matches the active Render Pipeline of the project:

- use the **URP version only in URP projects**;
- use the **HDRP version only in HDRP projects**.

Using the wrong pipeline version can prevent the effect from rendering correctly and may produce shader or VFX Graph errors.

Energy Flow also requires the **Visual Effect Graph** package.

If the project configuration is incomplete, open the **VFX Setup Assistant** and resolve the items under **Project-Wide Settings**.

Freshly imported VFX Graph assets can occasionally require recompilation. Use **Recompile VFX Graphs** under **Useful Tools** in the Setup Assistant before manually modifying the supplied graphs.

Play On Enable Is Off, but the Effect Is Still Visible in Edit Mode

This is expected.



Play On Enable controls runtime playback only.

The supplied Visual Effect uses its `OnPlay` Initial Event for automatic Editor preview, so the effect can continue to animate while working in Edit Mode even when Play On Enable is disabled.

When Play Mode begins:

- **Play On Enable = On** → the controller automatically starts the effect.
- **Play On Enable = Off** → the controller stops automatic runtime playback and waits for `Play()` to be called manually.

Surface Cannot Be Baked

Meshes used by the Surface Sample Baker must have **Read/Write** enabled.

If one of the selected Mesh or Skinned Mesh assets is not readable, the Baker stops and reports which mesh must be changed.

Also verify that:

- a valid Root is assigned;
- at least one Renderer is selected;
- the selected geometry contains triangles.

Surface Endpoint Does Not Work

A Surface endpoint requires both:

1. a valid **Surface Sample Asset** created by the Surface Sample Baker;
2. a **Flow Surface Source** using that asset.

If the Flow Surface Source cannot initialize, the Console reports problems such as a missing Sample Asset, missing proxy mesh, invalid sample data, missing compute shader, or incompatible bone mapping.

For Skinned Meshes, place the Flow Surface Source on the same Root hierarchy used when the surface was baked. Bone paths stored in the Surface Sample Asset are resolved relative to that Root.

Skeletal Animation and Blend Shapes

Surface endpoints support deformation produced by **skeletal animation**.

However, the current baked proxy does **not copy Blend Shapes**. If the source renderer uses Blend Shapes, those deformations are not represented by the Surface endpoint. The Baker reports a warning when such renderers are detected.

If the mesh topology, selected renderers, or skeletal hierarchy used by a Surface changes, bake a new Surface Sample Asset.

Sweep Appears to Do Nothing

Sweep behavior depends on the Start endpoint type.

Point: Sweep is disabled and behaves as Instant.

Line: Sweep uses the line's world X and Y position. If the line has no extent on the selected axis, every particle receives the same Sweep delay instead of producing a directional progression.

For example, Left Right does not create a directional progression on a perfectly vertical Line because all positions have the same world X coordinate.

Surface: Sweep uses normalized coordinates stored from the baked Surface.

Launch Does Nothing

Spawn Kick is a Surface-specific Launch mode.

It requires:

Start Mode = Surface and **Launch Mode = Spawn Kick**

Kick Speed uses the sampled surface normal, while Inherit Surface Velocity uses the available velocity of the source surface.

Launch has no effect when the Start endpoint is Point or Line.

Pre Fade Duration Uses a Normalized Value

Pre Fade Duration is a normalized `0–1` value relative to the particle's transfer delay, rather than a duration in seconds.

`0` keeps the particle invisible until the transfer begins.

`1` allows the fade-in to span the entire waiting period.

For example, with a 2-second delay and a Pre Fade Duration of `0.25`, the fade-in occurs during the final 0.5 seconds of the delay.

UFlow Preset Does Not Change Start or End

This is intentional.

UFlow presets preserve scene-specific Endpoint references. Applying a preset changes Playback and Flow Layer configuration but does not replace Point, Line, or Surface bindings.

Textures Are Missing After Moving a UFlow Preset

A normal **Save As..** preset references textures from the current Unity project and is intended to remain inside that project.

For transfer between projects, use **Export Preset..** and move both the `.uflow` file and its matching `_Res` folder.

Portable texture resources are resolved from that matching resource folder.

New presets should use normal project `Texture2D` assets instead of Unity's shared built-in texture container. Built-in references cannot always identify the original texture reliably when moved between projects.

Technical Reference: Parameter Glossary

This section describes the configurable parameters available in the **Energy Flow Controller Inspector**.

Parameters are grouped in the same categories used by the Controller. **Layer A, Layer B, and Layer C use the same parameter structure**, so the layer parameters are described only once.

Controller

General references and playback controls for the complete Energy Flow system.

- `Visual Effect` — Visual Effect component controlled by Energy Flow.

UFlow Preset

- `UFlow Preset` — UFlow preset assigned to the Controller.
- `Apply` — Applies the selected UFlow preset.
- `Save As..` — Saves the current configuration as a UFlow preset.
- `Export Preset..` — Exports a portable UFlow preset with texture resources.
- `Reset` — Restores the supplied default Energy Flow configuration.

Playback

- `Play On Enable` — Automatically starts the effect when enabled during Play Mode.
- `Mode` — Selects Infinite or Timed playback.
- `Duration` — Emission duration used by Timed playback.

Flow Layer

Each Flow Layer has its own independent settings.

- `Enabled` — Enables or disables the selected Flow Layer.

Spawn

Controls how particles are generated and distributed over time.

- `Spawn Mode` — Selects Variable, Periodic Burst, Single Burst, or Constant spawning.
- `Sweep Mode` — Controls directional transfer delay across Line or Surface endpoints.
- `Sweep Duration` — Controls how long the Sweep progression takes.

Variable

- `Rate Range` — Minimum and maximum variable emission rate.
- `Change Period Range` — Time range between emission-rate changes.

Periodic Burst

- `Count Range` — Particle count range for each repeating burst.
- `Delay Range` — Delay range between repeating bursts.

Single Burst

- `Count Range` — Particle count range for the single burst.
- `Delay Range` — Delay before the single burst.

Constant

- `Constant Rate` — Fixed continuous particle emission rate.

Endpoints

Defines where particles begin and where they travel.

Start

- `Mode` — Selects Point, Line, or Surface Start mode.
- `Point` — Transform used as the Start point.
- `Line Start` — First Transform defining the Start line.
- `Line End` — Second Transform defining the Start line.
- `Surface` — Flow Surface Source used as the Start surface.

End

- `Mode` — Selects Point, Line, or Surface End mode.
- `Point` — Transform used as the End point.
- `Line Start` — First Transform defining the End line.
- `Line End` — Second Transform defining the End line.
- `Surface` — Flow Surface Source used as the End surface.

Transfer

Controls particle travel between the Start and End endpoints.

Tracking

- `Start Tracking` — Selects Frozen or Live Start tracking.
- `End Tracking` — Selects Frozen or Live End tracking.
- `Speed` — Base particle transfer speed.

- `Start Delay` — Delay before transfer movement begins.
- `Pre Fade Duration` — Portion of the transfer delay used for fade-in. **Effective range: 0–1.**

Follow

- `Path Follow` — Controls how strongly Motions affect the followed path.
- `Loose Time` — Controls initial correction time after Spawn Kick.
- `Arrival Start` — Progress where final arrival correction begins.
- `Final Time` — Controls correction time near the target.

Launch

- `Mode` — Selects None or Spawn Kick.
- `Kick Follow Enabled` — Allows Spawn Kick particles to return toward the calculated path.
- `Kick Speed` — Initial speed along the Start Surface normal.
- `Inherit Surface Velocity` — Adds velocity inherited from the Start Surface.

PRE

PRE modifies the data used by the procedural Motions.

Progress

- `Progress Warp` — Remaps progress used by procedural Motions.
- `Progress Steps` — Quantizes Motion progress into discrete steps.

Window

- `Window Start` — Defines the beginning of the active Motion region.
- `Window End` — Defines the end of the active Motion region.
- `Window Softness` — Controls smoothing at the Window boundaries.
- `Window Peak Position` — Defines where the shared Motion envelope reaches maximum strength.

Modulation

- `Frequency Over Progress` — Modulates supported Motion frequencies along the path.
- `Phase Over Progress` — Modulates supported Motion phases along the path.

Variation

- `Strength` — Controls per-particle procedural Motion variation.

Animation

- `Time Scale` — Scales animation time used by procedural Motions.
- `Strobe Rate` — Quantizes Motion animation into discrete time steps.

Motions

Procedural Motions generate offsets from the direct Start-to-End path.

Bend

- `Path Position A` — Position of the first bend along the path.
- `Amount A` — Strength of the first bend.
- `Width A` — Width of the first bend region.
- `Angle A` — Direction of the first bend.
- `Path Position B` — Position of the second bend along the path.
- `Amount B` — Strength of the second bend.
- `Width B` — Width of the second bend region.
- `Angle B` — Direction of the second bend.

Oscillation

- `Amplitude A` — Amplitude of the first oscillation component.
- `Frequency A` — Frequency of the first oscillation component.
- `Phase A` — Initial phase of the first oscillation component.
- `Axis Angle` — Rotates the oscillation axes around the path.
- `Amplitude B` — Amplitude of the second oscillation component.
- `Frequency B` — Frequency of the second oscillation component.
- `Phase B` — Initial phase of the second oscillation component.
- `Animation Speed` — Controls oscillation animation speed.

Spiral

- `Radius` — Distance of the spiral from the base path.
- `Turns` — Number of spiral turns along the path.
- `Phase` — Initial angular offset of the spiral.
- `Rotation Speed` — Controls spiral rotation speed.

Lightning

- `Amount` — Strength of the jagged path displacement.
- `Segments` — Number of segments forming the lightning shape.
- `Refresh Rate` — Controls how frequently the lightning shape changes.
- `Seed` — Changes the generated lightning variation.

Organic Flow

- `Strength` — Strength of the organic path deformation.
- `Scale` — Frequency of organic noise variation along the path.
- `Animation Speed` — Controls organic noise animation speed.
- `Seed` — Changes the generated organic pattern.

Ribbon Weave

- `Radius` — Distance of strands from the central path.
- `Turns` — Number of weave rotations along the path.
- `Strand Count` — Number of available procedural strand positions.
- `Rotation Speed` — Controls weave rotation speed.

Branch Split

- `Spread` — Controls separation between procedural branches.
- `Branch Count` — Number of available procedural branch directions.
- `Twist Turns` — Number of twists applied to the branches.
- `Shape Power` — Controls where branch separation concentrates along the path.

Turbulence

- `Amount` — Strength of turbulent path displacement.
- `Frequency` — Frequency of turbulence variation along the path.
- `Animation Speed` — Controls turbulence animation speed.
- `Detail` — Adds smaller-scale noise detail.

POST

POST modifies the combined result of all active Motions.

Shape

- `Amount Over Progress` — Controls combined Motion strength along the path.
- `Radius Over Progress` — Controls radial scale along the path.
- `Axis Scale` — Independently scales the two transverse path axes.

Source Expand

- `Expand End` — Progress where the normal Motion offset is fully restored.
- `Expand Strength` — Controls Motion suppression near the Start endpoint.

Target Collapse

- `Collapse Start` — Progress where target convergence begins.
- `Collapse Strength` — Controls Motion suppression near the End endpoint.

Pulse

- `Strength` — Controls Pulse deformation strength.
- `Frequency` — Number of Pulse cycles along the path.
- `Phase` — Initial Pulse phase offset.
- `Speed` — Controls Pulse animation speed.

Roll

- `Angle` — Initial rotation around the path direction.
- `Speed` — Controls Roll rotation speed.

Appearance

Controls how the main Flow particles are rendered.

- `Texture` — Texture used by the Flow particles.
- `Size Range` — Minimum and maximum initial particle size.
- `Size Over Life` — Controls particle size throughout lifetime.
- `Color And Alpha Over Life` — Controls particle color and transparency throughout lifetime.
- `Color Intensity` — Multiplies particle color intensity.
- `Orientation Mode` — Selects Billboard or Along Velocity orientation.
- `Angle Z Range` — Random Z rotation range.
- `Smoothness` — Controls Lit surface smoothness.
- `Metallic` — Controls Lit surface metallic response.
- `Alpha Clipping` — Controls the alpha clipping threshold.
- `Fade Distance` — Controls soft fading near scene geometry.

Child

Each Flow Layer has its own independently configurable Child effect.

Trigger

- `Mode` — Selects when Child particles are generated.
- `Burst Count` — Number of particles generated by event triggers.
- `Rate` — Child emission rate used by Over Time.
- `Progress` — Parent progress threshold used by On Progress.
- `Spawn Chance` — Controls the probability or rate of Child generation.

Motion

- `Direction Mode` — Selects Random, Flow Direction, or Custom direction.
- `Reverse` — Reverses Flow Direction or Custom direction.
- `Custom Direction` — Direction vector used by Custom mode.
- `Random Direction From` — First boundary of the random direction range.
- `Random Direction To` — Second boundary of the random direction range.
- `Spread` — Adds randomized sideways directional variation.
- `Size Range` — Minimum and maximum initial Child size.
- `Lifetime Range` — Minimum and maximum Child particle lifetime.
- `Speed Range` — Minimum and maximum initial Child speed.
- `Gravity` — Acceleration applied to Child particles.
- `Drag` — Gradually reduces Child particle velocity.

Appearance

- `Texture` — Texture used by Child particles.
- `Size Over Life` — Controls Child particle size throughout lifetime.
- `Color And Alpha Over Life` — Controls Child color and transparency throughout lifetime.
- `Color Intensity` — Multiplies Child particle color intensity.
- `Orientation Mode` — Selects Billboard or Along Velocity orientation.
- `Smoothness` — Controls Lit surface smoothness.
- `Metallic` — Controls Lit surface metallic response.
- `Alpha Clipping` — Controls the alpha clipping threshold.
- `Fade Distance` — Controls soft fading near scene geometry.

Thank You!

I hope Energy Flow saves you time, gives you a flexible foundation for building energy-based VFX, and becomes genuinely useful in your projects.

If you have any questions, problems, or feedback, feel free to contact me.

Thank you for your support, and good luck with your project!

