

Simulation

Understanding the Simulation Loop

tyFlow's simulation loop is evaluated in the following way:

For each time step of the simulation:

- 1) The optimal order of events in the flow is found, by doing a depth-first search of connected events, starting with events that contain birth operators.
- 2) The ordered events are processed in ascending order, by evaluating the event's operators from top to bottom.

NOTE: Certain operators may have their evaluation deferred until later in the loop. Particle Physics operators set to "integration" mode are only evaluated during the bind solver step. Collision operators are also only evaluated after all other operators and non-PhysX solvers have been evaluated.

- 3) Once the events have been processed, the bind solver is evaluated. Cloth binds, particle binds, and any Particle Physics operators set to "integration" mode are evaluated.
- 4) Once the bind solver is finished, the wobble solver is evaluated. The wobble solver calculates spring forces applied to particles created with a Wobble operator.
- 5) Once the wobble solver is finished, the actor solver animates the transforms of any actor rig particles that have animation clips applied to them.
- 6) Once the actor solver is finished, all accumulated particle velocity and spin values are integrated into the system.
- 7) After velocity/spin integration, PhysX objects are processed. Internal PhysX rigidbodies have their position/velocities matched to their corresponding particles and the PhysX solver is evaluated.
- 8) Once the PhysX solver is finished, some final adjustments are made to the simulation. The bind solver checks to see if any particles should be put to sleep, the age of all particles is incremented, appropriate particles are cached, etc.

Simulation Validity

Under normal circumstances with moving particles, the simulation is history-dependent. This means that in order for the simulation state at time **[t]** to be known, the simulation state at time **[t - timestep]** must also be known. This also means that any given simulation state in those circumstances will only be valid for one timestep. As soon as the time changes, the simulation state changes, rendering the previous state invalid.

The exception to this case is if there are no changing properties in the simulation. If nothing is changing, then the simulation state at all times will be the same because the simulation is static.

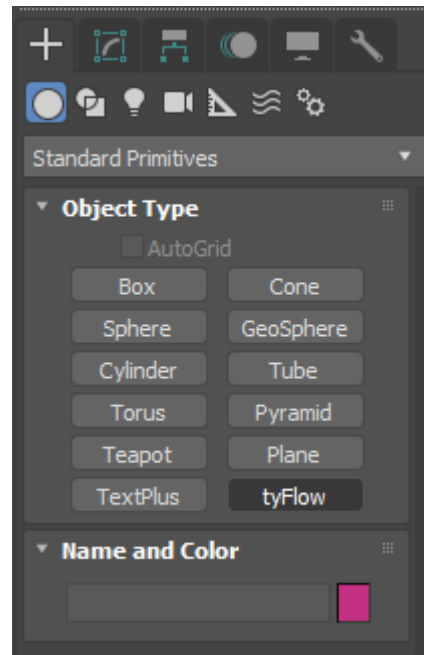
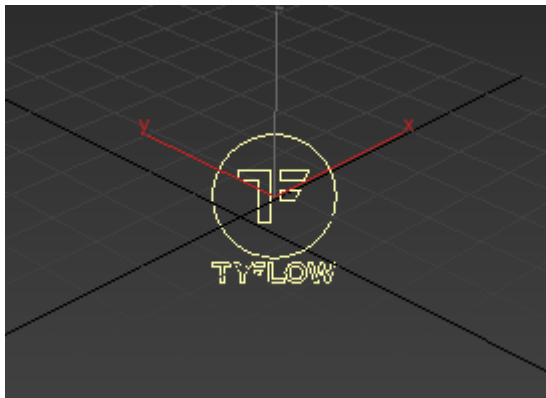
When the simulation is initialized, **tyFlow** iterates over all operators and queries them for the types of changes they will make to the simulation. If every single operator reports that it does not make changes to particles over time, then **tyFlow** will recognize that the simulation will remain static forever once all new particles have been birthed. This makes **tyFlow** a very efficient geometry scattering tool, since static particles that are scattered in a scene will not have to be continually evaluated after they are birthed.

NOTE: Grey text labeled: “Static: [range]” appears in the bottom left corner of the editor and displays info about which frames of the simulation are static, or tells you if the simulation is never static.

Getting Started

Creating a new tyFlow particle system

To create a **tyFlow** scene object, navigate to Create→Standard Primitives→**tyFlow** within the modifier panel and choose a location in the scene to place the object.



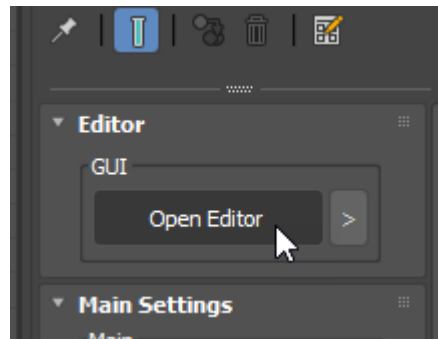
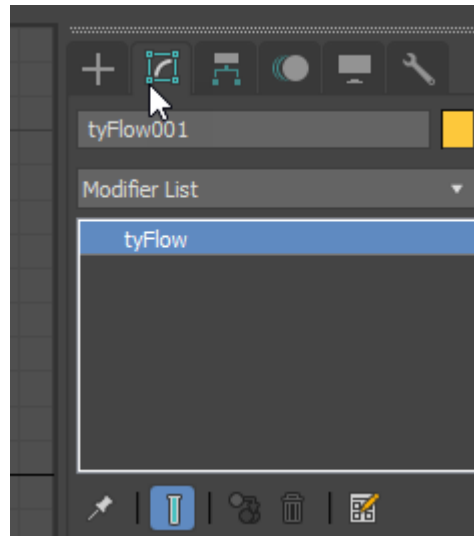
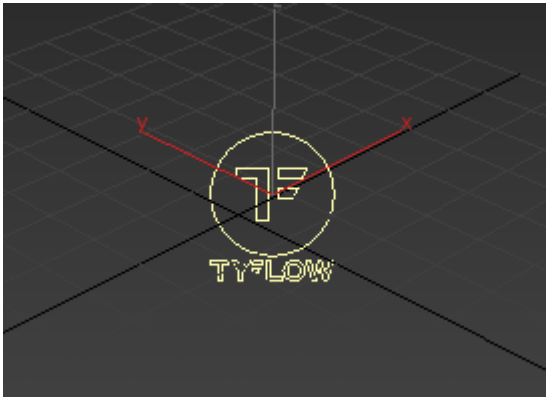
NOTE: The location you choose to place your **tyFlow** object does not matter – all **tyFlow** calculations happen in World-Space (not Object-Space), so the object's transform will have no effect on the simulation or display of the particles.

Using the Editor

The editor is where flows reside within a **tyFlow** object.

Opening the Editor

With the tyFlow icon selected in any viewport, go to the modifier panel and scroll down to open editor and click the button



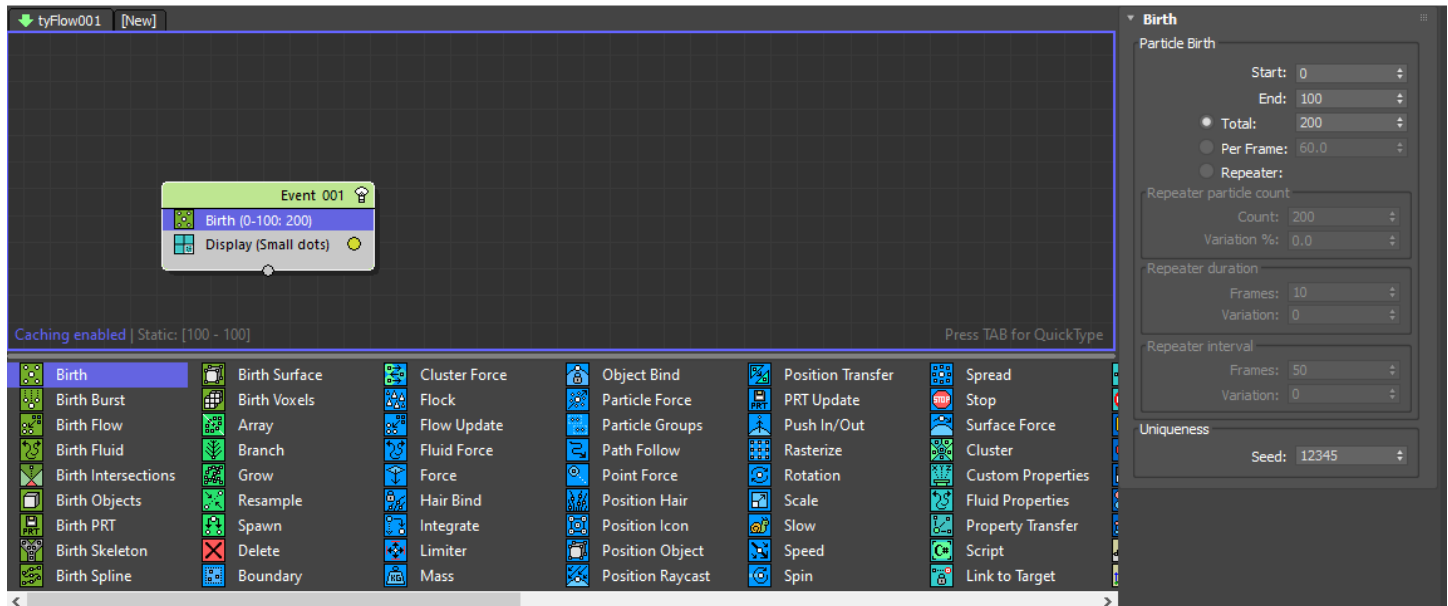
To open the editor for a particular **tyFlow** object, select ‘Open Editor’ from the modifier panel or “Editor...” for a particular object in the **tyFlow** viewport menu.

NOTE: Unlike *Particle Flow*, which features a single editor for all flows in the scene, **tyFlow** objects each have their own editor.

NOTE: The size and position of each editor window is saved to the scene file, and will be restored when loading the scene. If an editor is saved to an off-screen location, it will be automatically centered inside the current window

Navigating the Editor

By using your middle mouse button and scroll wheel, you can pan and zoom in the editor. Left-click can be used to select events/operators/connections/etc, and right-click can be used to display relevant context menus.



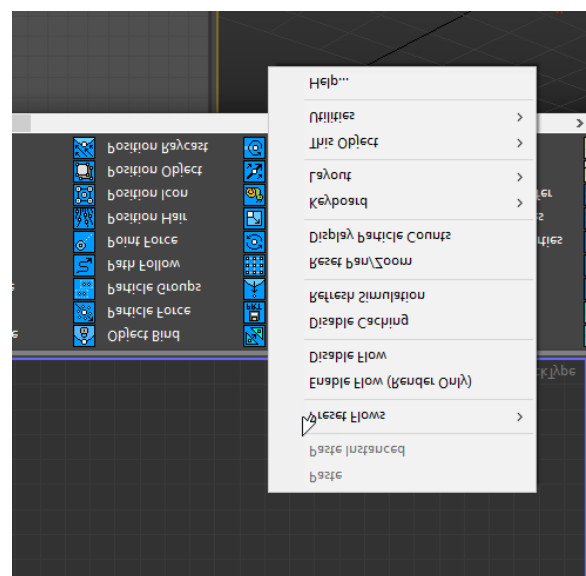
The grid view is where all flows are constructed. The operator list below the grid view displays all available operators. The rollout panel to the right of the grid view displays all settings for any operator that is selected.

NOTE: The pan and zoom value of each editor window is saved to the scene file, and will be restored when loading the scene.

Right-click Menus

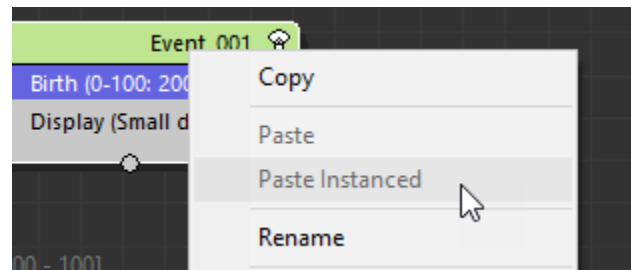
Various right-click context menus will appear when right-clicking the grid view, events, or operators.

While most right-click menu options are self-explanatory, a few may require some further details:

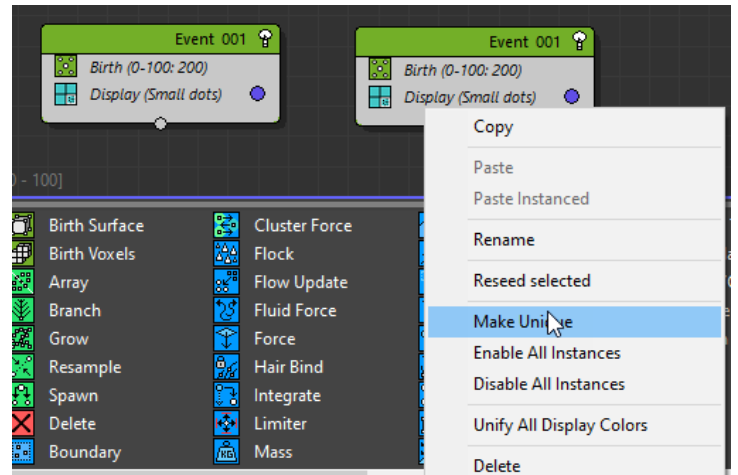


Operator right-click menus:

- **Paste Instanced:** operators that are pasted as instances will share all settings with the operator they were copied from. This menu item will only appear if an operator has previously been copied.



- **Make Unique:** choosing this option on an operator that is an instance of another operator will make its settings fully independent from that other operator. Therefore, from that point on it will no longer be an instance of the other operator. This menu item will only appear when right-clicking on an instanced operator.



- **Make Instances:** if multiple operators are selected, the operator that was right-clicked will become the data source of all the other operators which are selected. In other words, all operators that are selected will lose their current settings and be made instances of the operator for which this option is chosen. This menu item will only appear if multiple operators of the same type are selected.

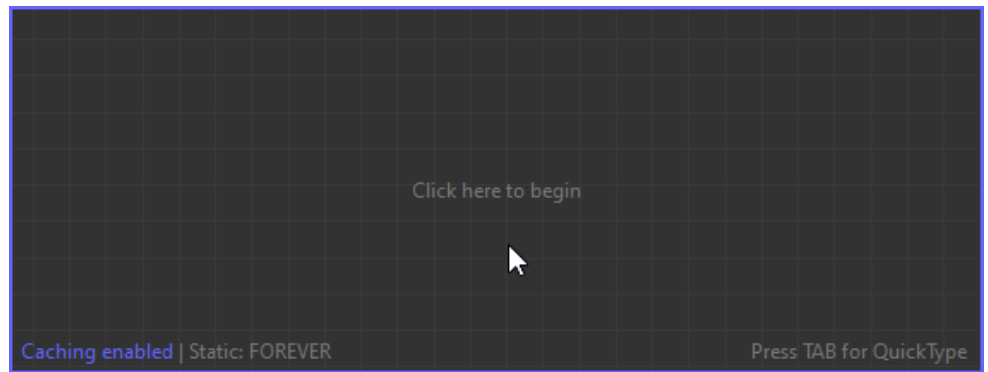
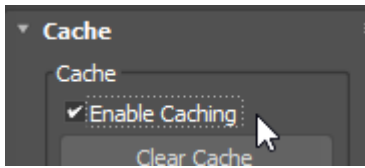
TIP

“Make instances” is an easy way to transfer settings between existing operators. For example, if you have two independent operators of the same type and you want them to have the same settings, instead of deleting one and copy/pasting the other in its place, simply select them both and choose “make instances” on the one whose data you want to copy to the other. The other operator will be instantly converted into an instance of the one you right-clicked. In that sense, “make instances” is shorthand for “make all of the other selected operators of the same type instances of this one”.

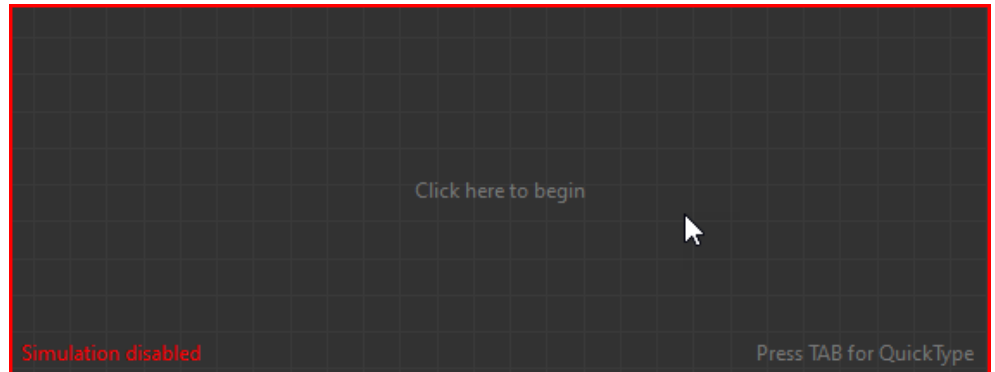
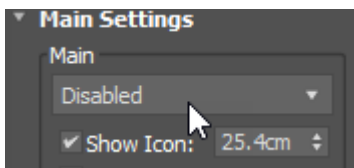
Editor hints

The editor displays various bits of information around its frame to assist users in diagnosing different behaviors of the flow.

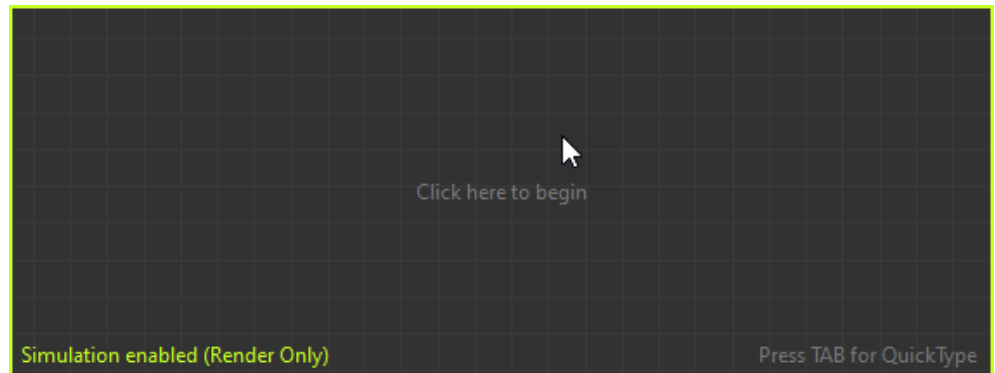
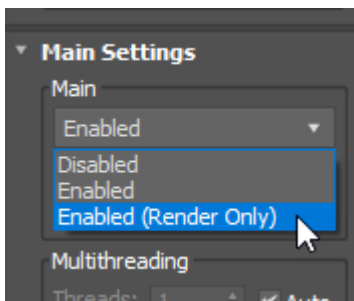
A **blue outline** around the frame of the editor means that realtime caching is enabled. "Caching enabled" will also appear in the bottom left corner.



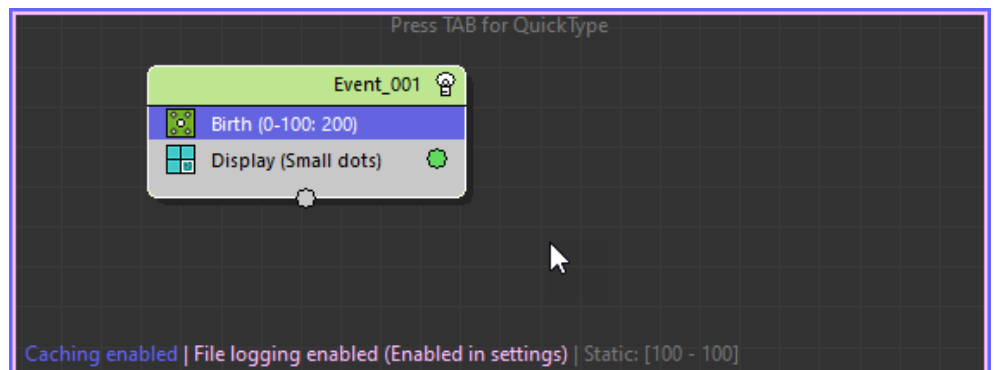
A **red outline** around the frame of the editor means that the flow has been disabled. "Simulation disabled" will also appear in the bottom left corner.

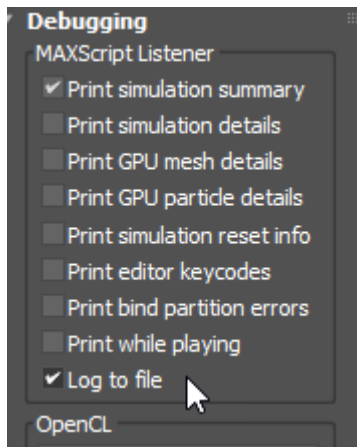


A **yellow outline** around the frame of the editor means that the flow is in render-only mode. "Simulation enabled (Render Only)" will also appear in the bottom left corner.

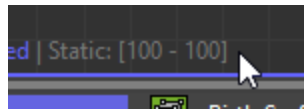


A **pink outline** around the frame of the editor means that the flow is logging its simulation progress to disk. Relevant information about the location of the log will also appear in the bottom left corner.





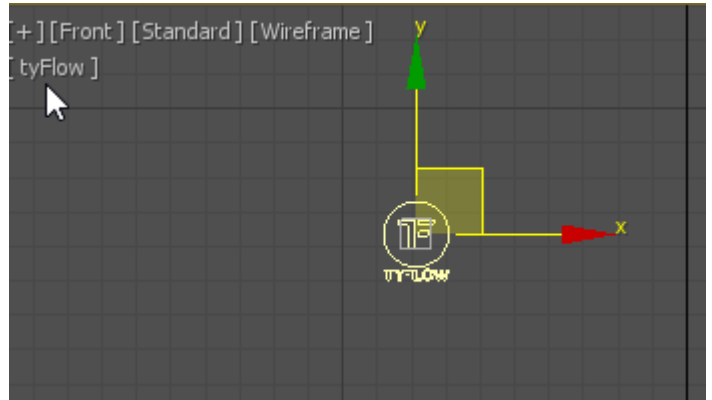
Grey text labeled “Static: [range]” appearing in the bottom left corner displays info about which frames of the simulation are static.



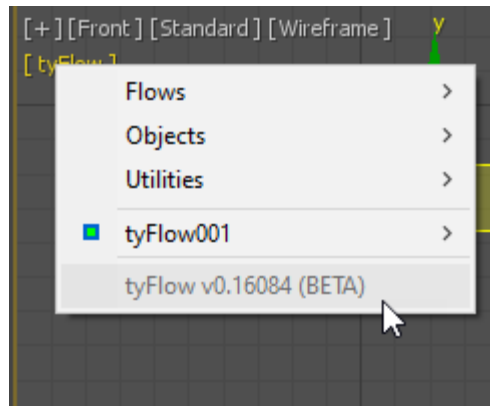
NOTE: A static frame is a frame which requires no additional simulation steps because it contains particles which do not change over time. A completely static flow with no moving particles will only need to evaluate a single simulation step, allowing for fast updates and timeline scrubbing.

Viewport Menu

tyFlow features a viewport menu (located in the top left corner of the active viewport) that will automatically be displayed when any **tyFlow** objects are present in the scene, and will be automatically hidden when no **tyFlow** objects are present in the scene. It can be used to quickly access and control available **tyFlow** objects, with options to enable/disable, hide/unhide, refresh, select and edit them. Options also exist to select/hide/unhide all existing **tyFlow** objects at once.



The viewport menu also displays the current version number of the **tyFlow** plugin file installed on the machine.



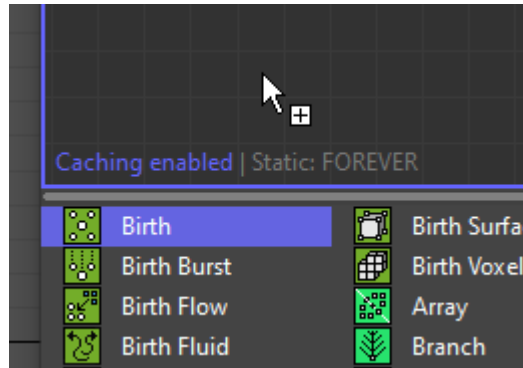
NOTE: Using the viewport menu to access **tyFlows** in the scene is often much easier than manually navigating to them through the viewport or scene explorer, and is part of an efficient workflow.

Creating Flows

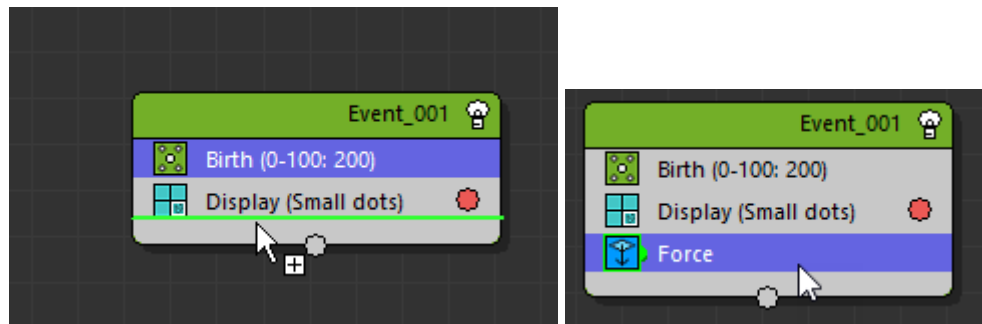
Flows are groupings of operators, events and connections between them. They allow you to direct the behavior of particles over time.

Creating Operators

To create a new operator, drag an operator from the operator list into the grid view.



An operator dragged into an existing event will be added to that event's operator list.



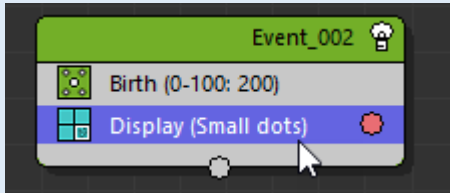
An operator dragged directly over the grid will be added to a new event. Operators can also be copied and pasted (from their right-click menu), and the same rules apply – operators pasted over the grid will be assigned to a new event, and operators pasted into an existing event will be added to that event's operator list.

NOTE: You can copy and paste multiple events and/or operators at the same time, by selecting them with the drag marquee, or by holding CTRL to individually select more than one at a time.

Creating Events

Events themselves cannot be created directly – only copied and pasted (from their right-click menu) or created by dragging an operator over the grid.

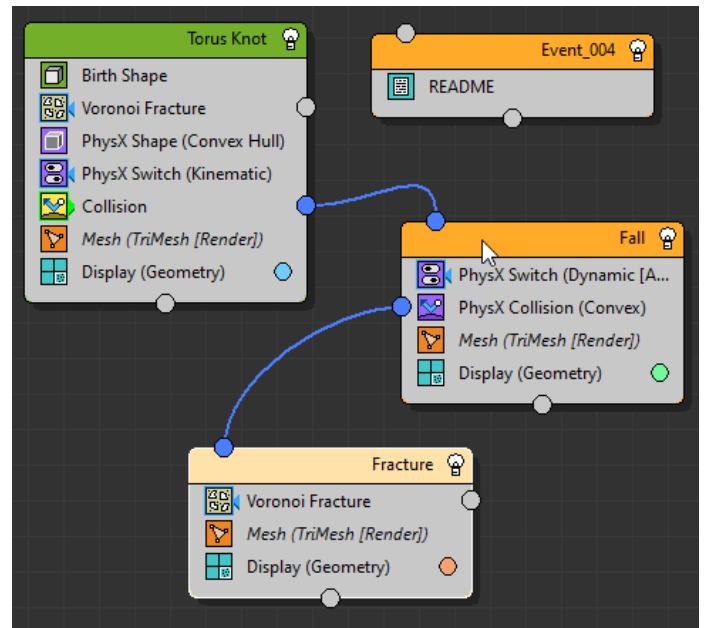
NOTE: When a new event is created it will automatically be assigned a new *Display* operator, if necessary.



Creating Connections

Connections between operators and events allow you to direct the behavior of particles over time. If an operator's output is connected to an event's input, any particle that satisfies the test condition of the operator will be sent to the connecting event at the end of the operator's simulation step. The direction of a flow is always forward (from operator to event) – operators can send particles to events, but events cannot send particles back into operators.

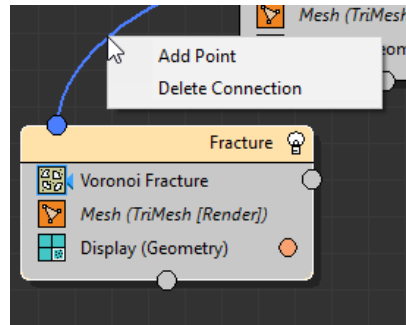
An event can take inputs from multiple operators, but an operator cannot output particles to multiple events.



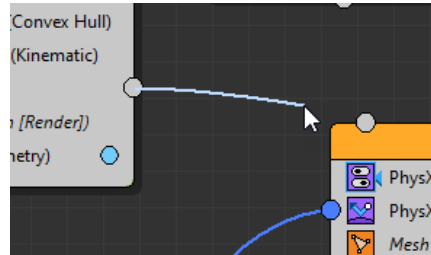
NOTE: **tyFlow** correctly handles event looping, where an operator is connected to a prior event, such that particles loop back to that same operator within the same timestep (normally resulting in an infinite loop that can never complete). No extra measures need to be taken to avoid infinite looping in those cases, as **tyFlow** will automatically ensure only a single loop is completed per timestep in such a scenario

Shaping connection wires

By right-clicking on a connection wire, you can add or remove points to it.



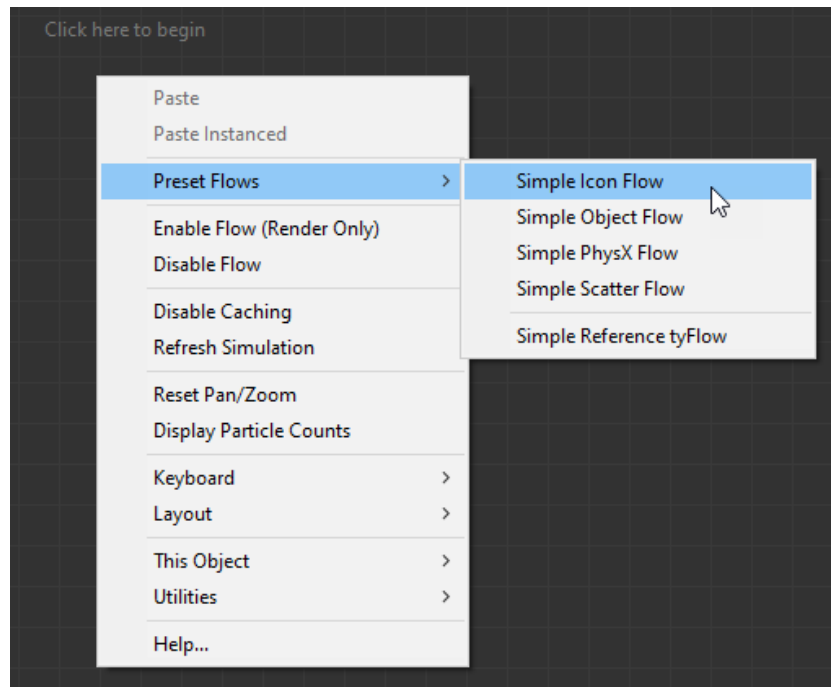
By dragging those points, you can shape the wire, allowing you to visually route wires around events in the grid.



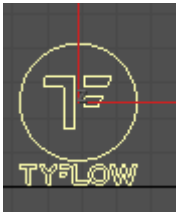
Preset Flows

In the right-click context menu of the grid view, you can find a “New” submenu that lists several preset flows which can be created.

Creating a new preset flow will *not* reset the editor or affect existing flows – it will merely *add* the selected preset flow to the editor. Relevant scene objects used to control preset flow properties will also be created, depending on the preset chosen.



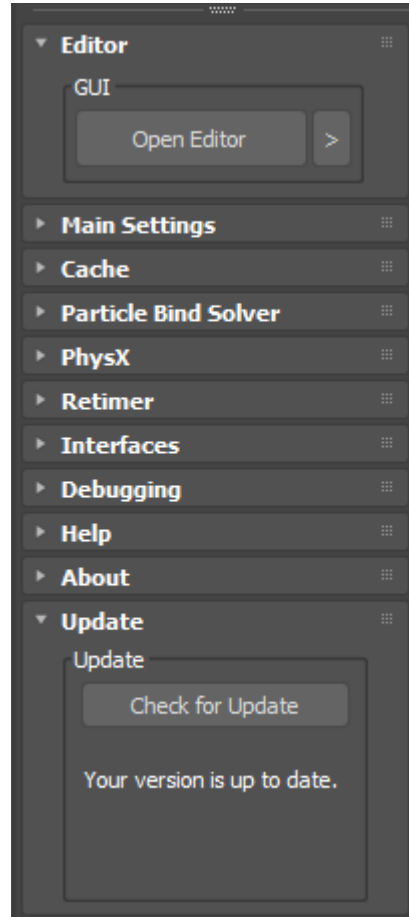
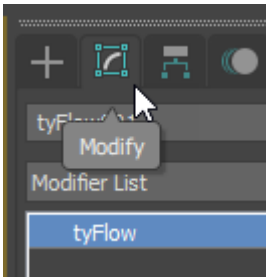
tyFlow Object Settings



tyFlow objects are what contain the events and operators requires to create and compute particle simulations.

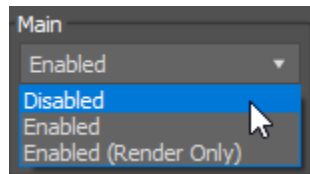
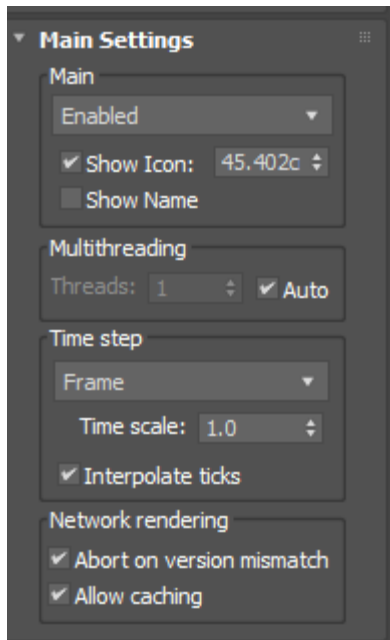
Each **tyFlow** object contains a variety of global settings which can be used to tune simulations, or export particles to various formats.

These settings can be accessed within the modifier panel.



We will go through each rollout category in the sidebar to find information about a particular setting.

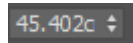
Main Settings Rollout



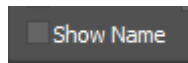
Enabled state dropdown: controls whether the flow is enabled, enabled (Render Only), or disabled. Disabled flows do not evaluate at all, while enabled (Render Only) flows only evaluate at render time.



Show icon: controls whether the **tyFlow** object's icon is visible in the viewport.

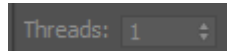


Icon size: controls the size of the **tyFlow** object's icon in the viewport.



Show name: controls whether the name of the **tyFlow** object is displayed in the viewport.

Multithreading



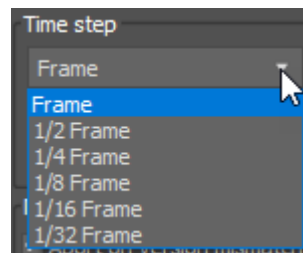
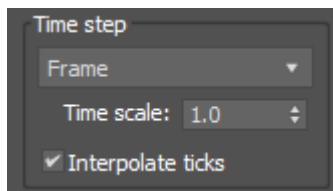
Thread count: controls the maximum number of CPU threads the flow can use to evaluate the simulation.



Auto: allows **tyFlow** to determine the maximum number of threads to use to evaluate the simulation (defaults to max. available)

Note: Setting thread count to a particular value doesn't mean that number of threads will be used for every operation, only that **tyFlow** may not exceed that particular number of threads for a given operation. Some operations benefit from more threads and some with less, and **tyFlow** makes internal determinations regarding the actual number of threads to use on a per-algorithm basis (with the only constraint being the maximum value provided by the user here).

Time Step



Time step dropdown: controls the number of steps the simulation will divide each frame into. With more time steps, simulation accuracy will be increased but simulation speed will be decreased. A value of 'Frame' is usually fine for simulations that don't require physical accuracy. A value of "1/4 Frame" or "1/8 Frame" is best for simulations featuring physically accurate constraints (sand/cloth/rope/etc) in order to increase overall simulation stability.

Time scale: 1.0

Time scale: the time scale setting allows you to control the speed of the simulation. Unlike the retimer settings, which allow you to control playback speed *after* the simulation is cached, the time scale setting allows you to control simulation speed *as it is calculated*. Values less than one have the effect of slowing the simulation, and values greater than one speed things up.

Note: Increasing the time scale value can lead to physics/PhysX inaccuracies, because increasing the value increases the velocity of all particles within the time step. You would only use this setting, as opposed to using the retimer, if you need to change the simulation speed of particles while maintaining the speed of all scene objects your particles are interacting with.

Interpolate ticks

Interpolate ticks: controls whether particle transforms will be interpolated at ticks between time steps.

Network rendering

Network rendering

Abort on version mismatch
 Allow caching

Abort on version mismatch: Rendering tyFlows on machines whose tyFlow version does not match the version of the originating scene file is usually a bad idea. While it doesn't guarantee problems, if the scene file relies on a feature not present in the version on the render machine, the scene may not render correctly. If this setting is enabled, renders will be automatically aborted on machines whose tyFlow version does not match the scene file, avoiding incorrect renders in the process.

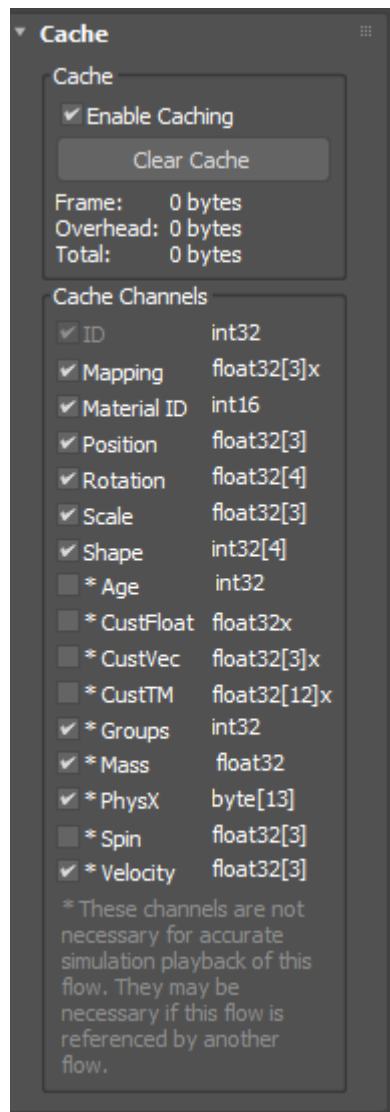
Note: It is always recommended to use the most recent version of **tyFlow**.

Allow caching: Controls whether render machines will cache frames while the simulation is processed on them prior to rendering. In most cases this should be left on, and only disabled if a particular render machine has enough RAM to process the simulation, but not enough RAM to cache each processed frame. Disabling this can greatly increase render time of complex simulations if multiple frames of the simulation are accessed in descending order (for example, in the case of a simulation retimed to play in reverse, or multiple flows referencing each other with staggered frame offsets).

Cache Settings Rollout

tyFlow's realtime timeline caching allows for smooth playback of a simulation in the viewport after it has been computed.

Note: Caches are saved on a per-frame basis, and do not adhere to the simulation's time step setting (the simulation itself will still run at whatever time step interval is chosen, but the cache will only save values once per frame). tyFlow will still interpolate the subframes of cache data if the flow's "interpolate ticks" setting is enabled, but if you need to playback your flow with sub-frame accuracy, you should disable caching.



Cache

Enable Caching: controls whether realtime timeline caching is enabled or disabled.

Clear Cache: clears the contents of the cache and effectively resets the simulation.

Cache channels

[Channel name - data type]: lists the available channels to save with the cache, and their corresponding data type.

Data types and their corresponding size in bytes:

byte = 1 byte
int16 = 2 bytes
int32 = 4 bytes
float16 = 2 bytes
float32 = 4 bytes

Note: The number in square brackets next to some data types represents the number of values that must be stored for that particular channel. For example, a position channel requires X/Y/Z values, each of which is stored as a float32 data type. So the size in bytes for a particular particle position value is 12 bytes (float32 x 3 values).

Mapping values are stored as a float32[3] x the number of mapping channels assigned to the particle.

Custom float data values are stored as a float32 x the number of custom float data channels assigned to the particle.

Custom vector data values are stored as a float32[3] x the number of custom vector data channels assigned to the particle.

Custom TM data values are stored as a float32[12] x the number of custom TM data channels assigned to the particle.

TIP:

You can estimate how much RAM will be required to cache a particular flow by using the following equation:

(number of particles) x (number of frames) x (total size of all saved channels in bytes)

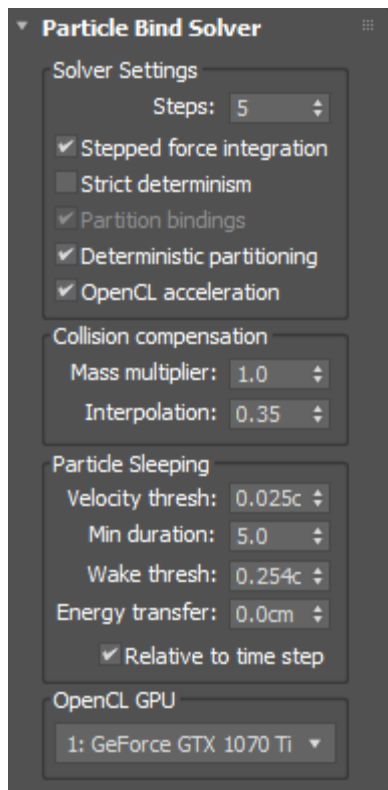
For example, a simulation of 1 million particles over 250 frames which caches particle positions and velocity will require approximately 7GB of RAM. This resulting value should only be considered an estimate, since caching requires some extra overhead not contained within any particular channel. The amount of overhead varies depending on the complexity of the flow and is displayed in the Cache rollout.

WARNING:

Caching can use a lot of RAM up very quickly, depending on the complexity of your flow. If your machine has limited RAM and you are planning on simulating many millions of particles, it is best to turn caching off and instead export the particles to disk using **tyFlow's** available export options. **tyFlow** makes **no** effort to check whether RAM allocations are possible on a given system, which means that if your RAM is full and **tyFlow** needs more of it in order to continue a simulation, **tyFlow** could cause crash. This 'fast and loose' approach to simulating is by design, and it is up to the user to ensure their system is capable of handling the simulations they are trying to run.

Particle Bind Solver Settings Rollout

tyFlow's particle bind solver is what solves all inter-particle bindings (aka **constraints** or **joints**) within tyFlow (excluding PhysX bindings). At its core, a binding is just a relationship between two particles, and solving many bindings in succession is what gives rise to intricate behaviors seen in materials like dirt, wet sand, cloth, ropes, etc. Proper tuning of the bind solver is important when simulating these complex systems.



Solver Settings

Steps: 5 **Steps:** controls the number of sub steps per simulation time step to solve all active bindings.

Note: The total number of evaluations per binding per frame can be calculated as (bind steps) x (simulation time steps). The higher the total number of evaluations, the more accurate the solver results will be. For granular simulations, a simulation time step of either “ $\frac{1}{4}$ Frame” or “ $\frac{1}{8}$ Frame” with bind solver steps of 5-10 is often adequate. For hires cloth simulations, bind solver steps may need to be much higher in order to maintain cloth stiffness.

Stepped force integration **Stepped force integration:** controls whether particle velocities are smoothly added to the bind solver per step, or added only once prior to all bind solver steps. Keeping this enabled has a very minor performance impact but can reduce high velocity artifacts in the resulting simulation.

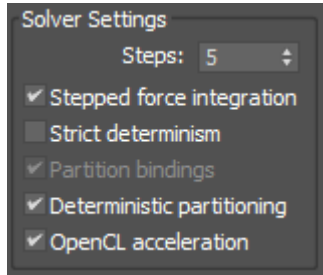
Strict determinism **Strict determinism:** with this setting turned on, successive runs of a simulation should return identical results. With this setting off, there is no guarantee that bindings will be evaluated in the same order or that race conditions between multiple threads will be prevented, and so results across multiple simulations may vary. Turning this setting on can have a detrimental performance impact, so it's recommended to keep it off, unless you need the simulation to produce identical results across successive runs.

WARNING: If you plan on rendering across multiple computers, “strict determinism” must be enabled or else the frames returned by different machines will not be in sync. An alternative to rendering with determinism on is to instead cache out your particles locally and then render a tyCache/PRT/etc loader instead of the tyFlow object itself.

If you choose to render your tyFlow with “strict determinism” on across multiple machines, make sure all machines have consistent OpenCL support. If you have OpenCL acceleration enabled but not all machines that you are using support OpenCL, mixing CPU/GPU solvers can impact determinism *even with “strict determinism” enabled*.

TIP:

It is generally a better practice to render a cache of your flow instead of your **tyFlow** object itself, when rendering across multiple machines. Rendering a cache ensures that hardware differences between computers have no impact on the consistency of the final output.



Solver Settings (continued)

Partition bindings: controls whether bindings will be split into non-overlapping groups, before being solved in a multithreaded manner. Turning this setting off can decrease simulation time, at the cost of increased error accumulation over time. This setting has no effect when OpenCL acceleration is enabled, because OpenCL acceleration requires partitions.

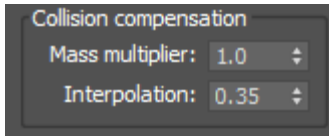
Deterministic partitioning: controls whether the partitioning process must avoid threaded race conditions. This setting can usually be disabled for granular flows, but should usually be enabled for cloth/soft-body flows to avoid jittering artifacts.

OpenCL acceleration: if an OpenCL2.0-compatible GPU device is found on the system, this option will be available. OpenCL acceleration can increase simulation performance, depending on the power of the available GPU. When enabled, all bindings will be solved on the GPU instead of the CPU.

Note: Enabling OpenCL acceleration does not guarantee a performance boost. There is a fair amount of overhead involved in transferring data to-and-from the GPU during the simulation, that can offset the actual speed boost the GPU offers during its calculation phase. While an overall increase in performance should be expected for very high-end GPUs on systems with few CPU cores, a system with many CPU cores and a low-end GPU may not see much of a performance boost with OpenCL at all. Results will vary across hardware and users must experiment to determine if OpenCL acceleration is right for them. It is not a magic bullet solution.

Collision Compensation

During each simulation step, collisions are always processed after bindings. No solid geometry collisions are processed while the bind solver evaluates bindings each bind solver step. Because of this, it's possible for the bind solver to pull particles straight through colliders, only for the subsequent collision step to fix those intersections afterwards. However, even though those intersections are eventually fixed, the rest of the bindings remain unaware that such a collision ever took place, and this can cause visual artifacts within the overall bind network. To compensate for this, particle masses can be artificially adjusted when collisions are detected on the previous simulation substep. Collided particles can be given a heavier mass, so that they won't be pulled as forcefully by their connected particles. Once previously-collided particles are determined to have no more collisions, their mass will return to normal. The combination of these effects can help reduce visual artifacts in binding networks (like cloth).

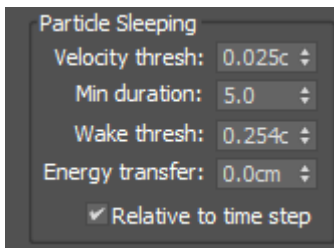


Mass multiplier: The multiplier applied to the (inverse) mass of particles that collided on the previous simulation step. The smaller the value, the less influence surrounding bindings will have on a collided particle.

Interpolation: The interpolation speed used to transition particle masses between the collision compensation value and their original value. Keeping this value low can help prevent jittering artifacts caused by the masses of collided particles switching between the compensation value and their original value too quickly.

Particle Sleeping

By enabling particle sleeping, you can force low-velocity particles to come to a standstill when they would otherwise keep moving over time. This can prevent unwanted motion in particles and forcibly bring jittering particles to rest.



Velocity thresh: particles whose velocity magnitude is below this threshold will be considered candidates for sleep.

Min duration: candidate particles whose velocity magnitude remains under the velocity threshold for this duration of time will be put to sleep.

Wake thresh: sleeping particles whose velocity exceeds this value at the end of a time step will be awoken.

Energy transfer: the amount of neighbor-particle energy that can contribute to waking a particle.

Relative to time step: Multiplies threshold velocities by the time step.

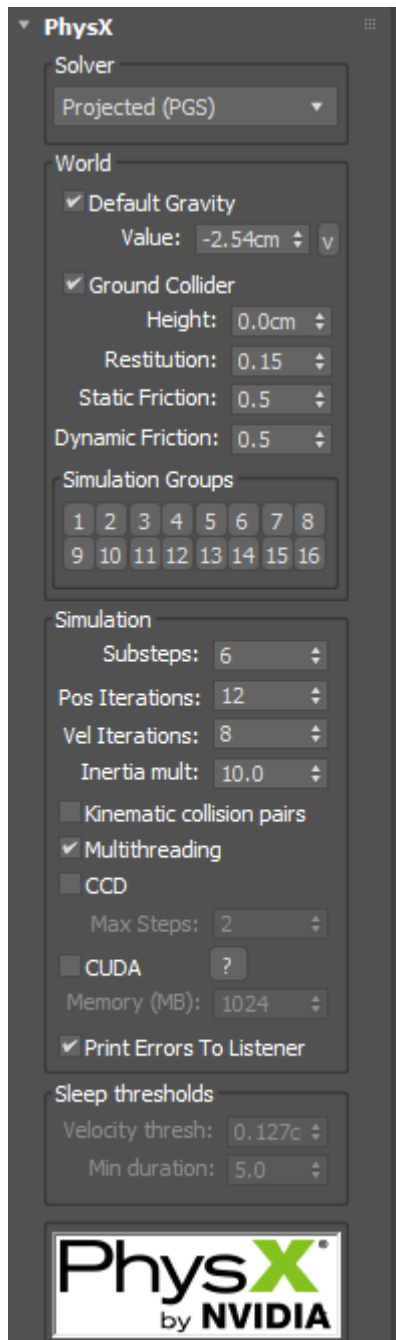
INFO:

Because velocities are integrated each time step, wake/sleep thresholds may be too large by default if your time step is less than 1. For example, if your gravity strength is -1.0 and your sleep threshold is 0.5, particles will not fall asleep when your time step is 1 frame. However, if your time step is $\frac{1}{2}$ frame, particles will fall asleep because at each substep their velocity is increased by 0.5 instead of 1.0 (which matches the sleep threshold). If “relative to time step” is enabled, the wake/sleep thresholds will be multiplied by the time step delta, and so in this example the effective threshold would actually be 0.25 ($0.5 * \frac{1}{2}$) per step.

NOTE: Particle sleeping has no performance impact. Its impact is purely visual. Sleeping particles will still be evaluated by the solver – the difference is that if they are considered asleep at the end of a time step, they will be returned to their previous location (effectively rendering them motionless).

PhysX Rollout

These controls affect all PhysX rigid bodies in the simulation.



Solver

PGS/TGS: controls which PhysX solver to use in the simulation

INFO: Information about each solver (Projected/Temporal Gauss-Seidel) can be found on NVidia's PhysX website. TGS is a relatively new addition to PhysX, and is typically faster than PGS for rigid body simulations, but can produce unexpected results when solving PhysX bindings (constraints). In general, it should be treated as an experimental solver, and PGS should still be used by default.

World

Default gravity: controls whether a default gravity force will be applied to all PhysX particles.

Gravity value: controls the strength of the default gravity force.

Ground collider: controls whether a default ground collider will be added to the PhysX simulation.

Height: the height of the default ground collider, in world-space.

Restitution: the restitution of the default ground collider.

Static friction: the static friction of the default ground collider.

Dynamic friction: the dynamic friction of the default ground collider.

Simulation groups: the simulation groups that will be affected by the ground collider

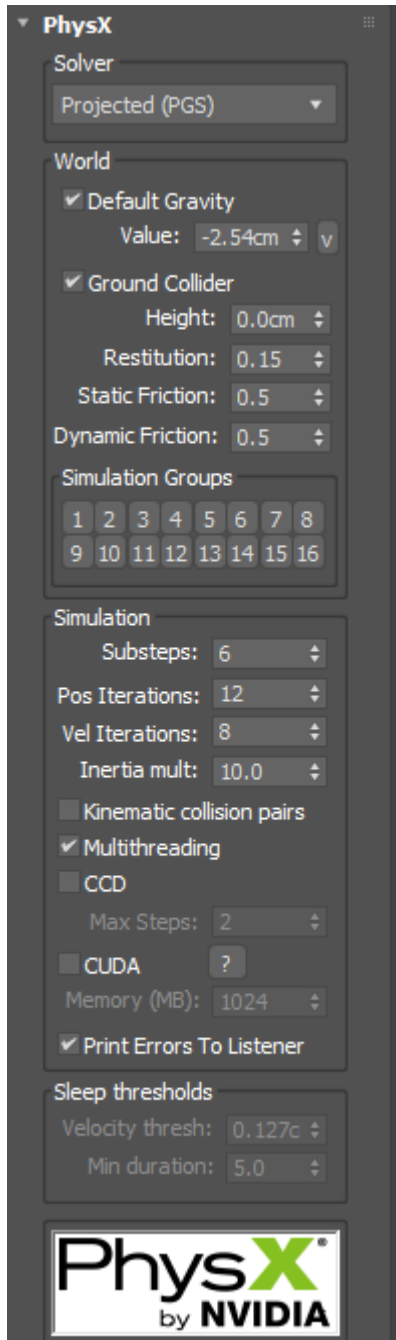
Simulation

Substeps: the number of substeps that will be calculated per time-step for the PhysX simulation.

TIP:

Increase substeps to increase the overall accuracy of the simulation. For high-fidelity simulations, a value of 12 or higher may be more appropriate than the default value.

PhysX Rollout (continued)



Simulation (continued)

Pos iterations: the number of position iterations that will be used to solve joint/contact constraints, per rigidbody, per substep.

Vel iterations: the number of velocity iterations that will be used to solve joint/contact constraints, per rigidbody, per substep.

TIP:

Increase pos/vel iterations to reduce jittering when simulating particles with PhysX bindings.

Inertia mult: this is a multiplier which affects all PhysX particles' inertia. Higher values can increase simulation stability while decreasing angular acceleration/deceleration. Lower values increase angular acceleration/deceleration, but can lead to jittering or other instabilities. For smaller objects with low mass, this value can be lowered (0.5 - 1.0). For bigger objects with a lot of mass, it should be kept high (5.0 - 20.0).

Kinematic collision pairs: controls whether inter-penetrating particle rigidbody pairs that are set to 'kinematic' or 'trigger' will generate contacts.

Multithreading: controls whether the PhysX engine will make use of multiple CPU threads.

CCD: controls whether *continuous collision detection* is enabled or disabled. *Continuous collision detection* can prevent collision tunnelling between high-velocity rigidbodies, for a performance cost.

CCD steps: controls the number of substeps used to resolve collisions by the CCD engine.

CUDA: controls whether PhysX computations will be accelerated with CUDA.

INFO:

In order for CUDA acceleration to work, tyFlow requires that two DLL files (PhysXDevice64.DLL and PhysXGPU64.DLL - both available on the tyFlow download page) be placed in the same folder where the tyFlow DLO file is loaded from.

NOTE: CUDA acceleration does not always guarantee faster simulations. Due to performance costs related to transferring necessary data to-and-from the GPU, speed benefits from CUDA might not be apparent until hundreds/thousands of rigid body particles are in the simulation. When a simulation has only a few rigid body particles, CUDA acceleration being enabled may actually decrease overall performance.

Memory (MB): 1024

Memory MB: the amount of GPU memory to allocate for CUDA computations.

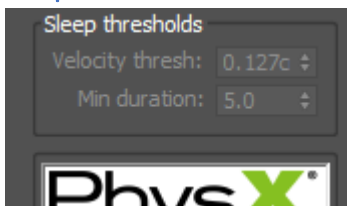
INFO:

Setting the CUDA memory limit higher than the default value does not mean the simulation will run faster. The memory limit controls the amount of VRAM to allocate for constraint/contact processing, and generally a CUDA simulation does not require much VRAM in order to process all contacts, even if a lot of rigid bodies are present in the simulation. Setting this value very high is usually unnecessary, and can actually contribute to slowdowns at the beginning of the simulation, due to the time it takes to initially allocate the VRAM. Just because you have a GPU with a lot of VRAM, does not necessarily mean you should increase this setting from its default value. For reference, the default value suggested by NVidia is approximately 140mb.

Print Errors To Listener

Print errors to listener: controls whether to print PhysX simulation errors (reported internally by the PhysX engine) to the MAXScript listener.

Sleep thresholds



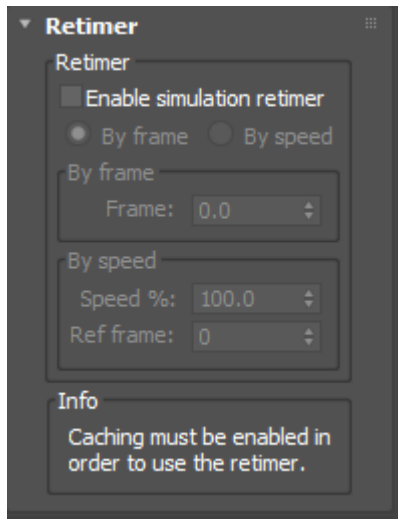
Velocity thresh: Particles with a linear/angular velocity below this threshold at the end of the time step will be candidates for sleeping.

Min duration: Particles which satisfy the velocity threshold for this number of frames will be put to sleep.

NOTE: Strange behavior can occur if particles with PhysX Bindings are put to sleep, therefore particles with PhysX Bindings will ignore the sleep threshold settings.

Retimer Rollout

The **tyFlow** retimer can be used to retime an entire simulation. By animating the retimer playback frame value, you can choose what part of the simulation plays back over the course of the timeline. Subframe values will be correctly interpolated by the retimer, allowing for smooth slow-down or speed-up effects.



Enable simulation retimer: controls whether simulation playback will be controlled by the retimer frame value.

Retime type: controls whether the retimer affects playback frame or speed.

By frame

Frame: the retimer frame value that controls simulation playback. Animate this value to control the current playback frame.

By speed

Speed %: the percent value that controls simulation playback speed.

Ref Frame: the reference frame that the speed multiplier will be relative to.

INFO:

Setting a proper reference frame is important for the speed multiplier. The reference frame should typically be the start frame of your playback sequence, not necessarily the start frame of the simulation.

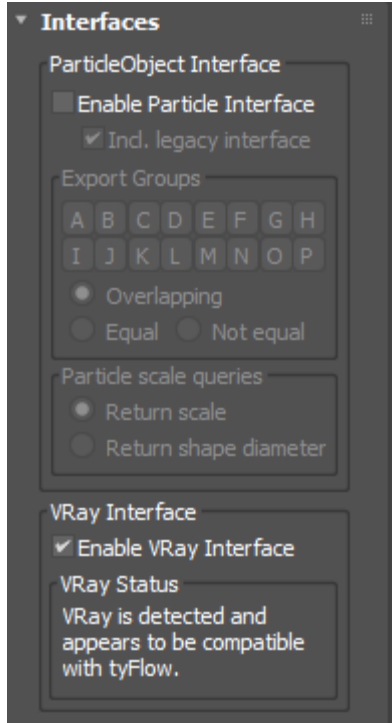
The equation for the speed retimer is:

$$\text{playbackFrame} = [\text{referenceFrame} + \text{abs}(\text{time} - \text{referenceFrame}) * \text{speed} * \text{sign}(\text{time} - \text{referenceFrame})].$$

TIP: The speed value is not animatable. If you need a variable playback speed, use the retimer in “frame” mode instead.

Interfaces

Each **tyFlow** is able to export “I_PARTICLEOBJ” and “I_VRAYGEOMETRY” interfaces to 3ds Max. In some cases, enabling these can cause issues between **tyFlow** and other plugins. In other cases, you might want to limit which particles are sent back to a particle query request. These settings allow you to override these interface exports.



ParticleObject Interface

Enable Particle Interface: controls whether the particle interface will be returned when the **tyFlow** object is queried for its default interface. Returning this interface allows third-party plugins to query a **tyFlow** for its particles.

Incl. legacy interface: when enabled, 3ds Max’s legacy particle interface (I_PARTICLEOBJ) will be enabled, as well as its modern interface (IParticleObjectExt).

Note: Disabling the “incl. legacy interface” setting can improve compatibility with certain 3rd party plugins.

Export groups: controls which particle export groups will be returned when a **tyFlow** is queried for its particles. Use these groups to limit which particles will be sent to third-party plugins querying a **tyFlow** for its I_PARTICLEOBJ interface.

TIP:

For example, if you have a PhoenixFD grid which uses a **tyFlow** as a PHXSource input object, you might only want to use certain particles in the flow to affect the PhoenixFD simulation. By limiting the particles exported through the I_PARTICLEOBJ interface (by setting the desired export groups), you can limit which particles will be returned to the PhoenixFD object.

Particle scale queries

When a 3rd party plugin asks tyFlow for the scale of a particle, these options let you choose what property of the particle is returned.

Return scale: the explicit scale value of the particle will be returned.

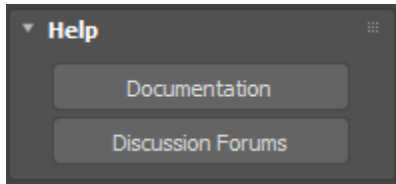
Return shape diameter: the diameter of the particle’s shape mesh will be returned.

VRay Interface

Enable VRay Interface: controls whether the I_VRAYGEOMETRY interface will be returned when the **tyFlow** object is queried by VRay for its VRay-compatible interface.

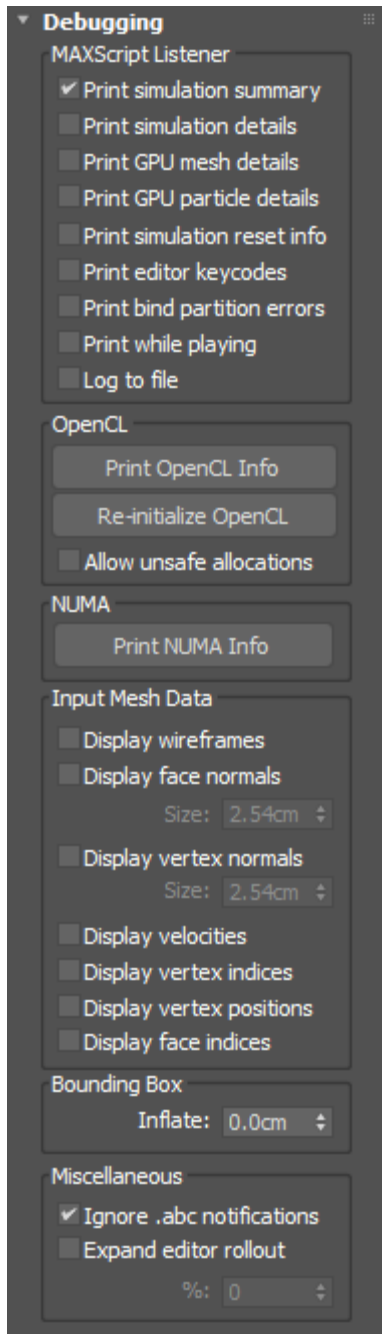
Help Rollout

The Help rollout contains links to the official **tyFlow** documentation, as well as the official **tyFlow** discussion forums.



Debugging Rollout

This rollout contains controls which allow users to profile and debug various aspects of a flow.



MAXScript Listener

Print simulation summary: controls whether a summary of flow details will be printed to the MAXScript listener each time a sequence of frames is simulated.

Print simulation details: controls whether a verbose list of simulation timing details will be printed to the MAXScript listener each time a sequence of frames is simulated.

Print GPU mesh details: controls whether a verbose list of timing details will be printed to the MAXScript listener regarding the time required to upload **tyFlow** meshes to the GPU for display.

Print GPU particle details: controls whether a verbose list of timing details will be printed to the MAXScript listener regarding the time required to upload **tyFlow** particle transforms to the GPU for display.

Note: Simulation details can provide useful insights into the time required to complete each step of the simulation process. If a simulation is running slowly, you can view the simulation details to see which operator or function is taking the most time to process.

Computing simulation details has a minor performance cost, and printing simulation details to the listener is a slow process, so these settings should remain disabled unless you are actively trying to debug or profile a simulation.

Print simulation reset info: controls whether information about changes to input objects is printed to the MAXScript listener.

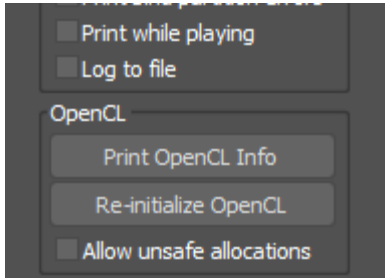
TIP:

Every time a flow's input objects change, the flow's simulation is reset to account for the changes. Sometimes a buggy input object may send rogue notifications to the flow, announcing it has changed, even though hit has not. This can cause a flow to continually reset its simulation in an undesirable manner. Enabling this setting can help users figure out which objects are sending change notifications to the flow.

Print editor keycode: keycodes of keys pressed in the editor will be printed in the MAXScript listener.

Print bind partition errors: if a bind partition contains adjacent constraints, an error will be printed to the MAXScript listener.

Note: Partition error printing is a developer setting that should generally be kept off - it involves extra calculations that will slow down the simulation, and doesn't provide any useful information to regular users.



Print while playing: controls whether simulation summaries or details will be printed to the MAXScript listener during timeline playback. Keeping this setting disabled will maximize viewport playback speed, by suppressing messages while playback is occurring.

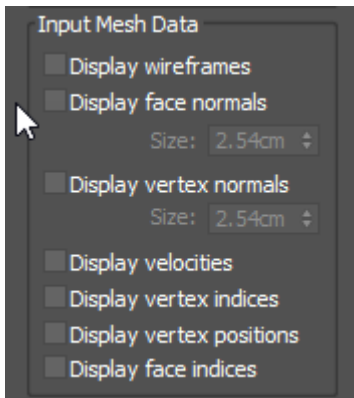
Log to file: simulation profiler data will be saved to a log file on the hard disk.

OpenCL

Print OpenCL Info: pressing this button will print a list of available OpenCL devices and their properties to the MAXScript listener.

Re-initialize OpenCL: rebuilds OpenCL programs and kernels. In case of an OpenCL error during simulation, OpenCL must be re-initialized before it can be utilized again.

Allow unsafe allocations: allows OpenCL to try to allocate more VRAM than the default allocation limit (which is $\frac{1}{4}$ of total VRAM), where necessary.



Input Mesh Data

These settings allow users to see visual properties of a flow's input meshes in the viewport. The resulting markers represent the exact data held in RAM by the flow in its custom **tyMesh** object format, not necessarily the raw data contained within the input objects' Mesh objects. Usually, there should not be a discrepancy between the two, but these options will allow you to see if there is.

Display wireframes: manually draws the wireframes of all input meshes in the viewport.

Display face normals: manually draws the face normals of all input meshes in the viewport.

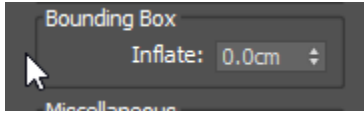
Face normal size: controls the overall length of the drawn normals.

Display velocities: manually draws vertex velocity vectors of all input meshes in the viewport.

Display vertex indices: manually draws vertex index numbers for all faces of all input meshes in the viewport.

Display vertex positions: manually draws vertex position values of all input meshes in the viewport.

Display face indices: manually draws face index numbers for all faces of all input meshes in the viewport.

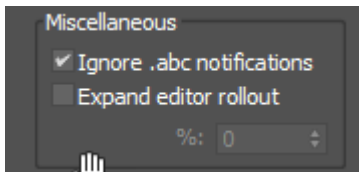


Bounding box

Inflate: manually inflates the bounding box of the **tyFlow** object by the specified value.

INFO:

By default, the bounding box of a flow encapsulates all of its particle positions. In order to maximize performance, the bounding box does not encapsulate the shape mesh of each particle, merely the particle's volumeless 3D position in space. Due to this optimization, if all of the particles of a flow are positioned outside of a viewport's frustum, the entire flow may be culled from display even if the shape meshes of particles are large enough to overlap the interior of the frustum. By manually inflating the bounding box of a flow to account for the size of its particle meshes, you can ensure that the flow will not be culled from viewport display even if no particle positions are in view.



Miscellaneous

Ignore .abc notifications: Ignores rogue change notifications sent by imported Alembic files.

INFO:

Max's default Alembic importer sends rogue change notifications to its dependent objects when the time slider is moved. Enabling this setting will ignore those notifications, and prevent **tyFlow** simulations which are dependent on Alembic objects from resetting each time the time slider is moved.

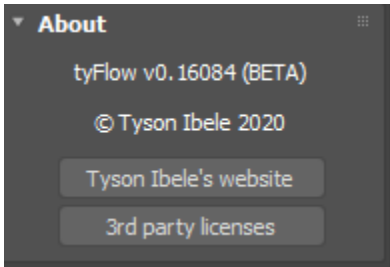
Expand editor rollout: some Windows display configurations can result in the **tyFlow** editor rollout being cropped improperly. Enabling this setting and increasing the percentage spinner will increase the width of the editor rollout.

%: the percentage (relative to default width) to increase the editor rollout.

Note: This is a sticky setting that will persist across max scene files.

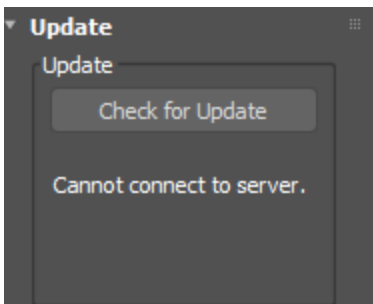
About Rollout

The About rollout contains copyright information, a link to Tyson Ibele's website, as well as third party license information.
























































































































Update Rollout

The Update rollout gives you the ability to check online to see if the installed version of **tyFlow** is older than the latest available version.



Operators

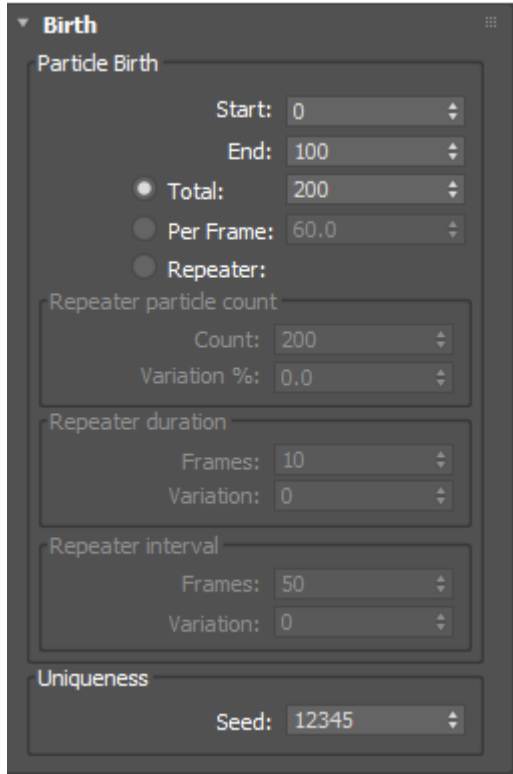
Each of these operators will be covered in the following pages:

 Birth	 Flow Update	 Rotation	 Vertex Color	 Cloth Bind	 Mesh
 Birth Burst	 Fluid Force	 Scale	 Displace	 Cloth Collect	 Spline Paths
 Birth Flow	 Force	 Slow	 Move Pivots	 Modify Bindings	 Collision
 Birth Fluid	 Hair Bind	 Speed	 Relax	 Particle Bind	 Find Target
 Birth Intersections	 Integrate	 Spin	 Shape	 Particle Break	 Object Test
 Birth Objects	 Limiter	 Spread	 Shape Remove	 Particle Physics	 Property Test
 Birth PRT	 Mass	 Stop	 Shell	 Particle Switch	 Send Out
 Birth Skeleton	 Object Bind	 Surface Force	 Smooth	 Wobble	 Split
 Birth Spline	 Particle Force	 Cluster	 Subdivide	 PhysX Bind	 Surface Test
 Birth Surface	 Particle Groups	 Custom Properties	 Tets	 PhysX Break	 Time Test
 Birth Voxels	 Path Follow	 Fluid Properties	 Weld	 PhysX Collision	 Camera Cull
 Array	 Point Force	 Property Transfer	 Brick Fracture	 PhysX Fluid	 Display
 Branch	 Position Hair	 Script	 Convex Fracture	 PhysX Modify	 Display Data
 Grow	 Position Icon	 Link to Target	 Edge Fracture	 PhysX Shape	 Notes
 Resample	 Position Object	 Move to Target	 Element Attach	 PhysX Switch	 Export Particles
 Spawn	 Position Raycast	 Set Target	 Element Fracture	 Actor	 Baseline
 Delete	 Position Transfer	 Instance ID	 Face Fracture	 Actor Animation	 Separator
 Boundary	 PRT Update	 Instance Material	 Fuse	 Actor Center	
 Cluster Force	 Push In/Out	 Mapping	 Voronoi Fracture	 Actor Collect	
 Flock	 Rasterize	 Material ID	 Angle Bind	 Actor Convert	

Birth operator



The Birth operator is the simplest operator for creating new particles over time.



Start/End: controls the time range in which to birth new particles.

Total: (Count) controls the total number of particles to birth over the time range.

Per frame: (Count) controls the number of particles to birth per frame over the time range.

Repeater: Repeater:

Repeater particle count

Count: controls the number of particles to birth per interval.

Variation %: the per-particle percentage of variation to apply.

Repeater duration

Frames: controls the number of active birth frames per interval, within the overall time range.

Variation: the per-particle amount of variation to apply.

Repeater interval

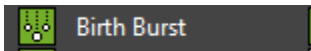
Frames: controls the amount of time to wait between birth intervals.

Variation: the per-particle amount of variation to apply.

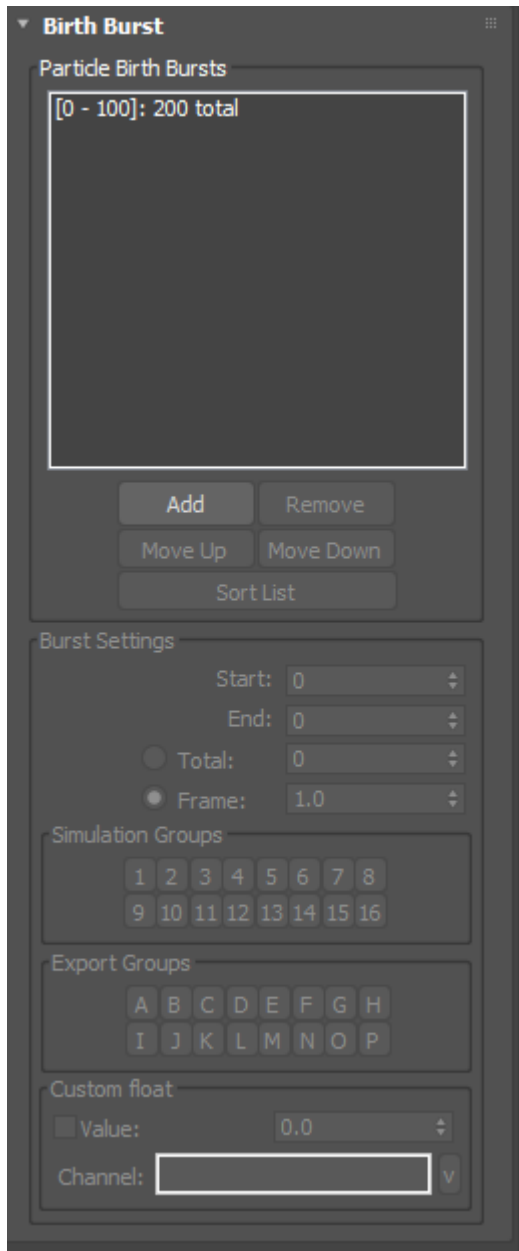
TIP

The repeater is useful for repetitive particle birth sequences. For example, imagine you wanted to shoot 50 particles into the scene, every 20 frames. You would enable the repeater and set *count* to 50, *duration* to 1 and *interval* to 20. Using the repeater avoids having to setup multiple birth operators to achieve the same outcome.

Birth Burst operator



The Birth Burst operator allows you to define multiple unique birth events, aka “bursts”.



Particle Birth Bursts:

Birth burst list: the list of all active birth bursts.

Burst Settings

Start/End: controls the time range in which the selected burst will birth new particles.

Total: (Count) controls the total number of particles the burst will create over the time range.

Frame (Count) controls the number of particles the burst will create per frame over the time range.

Simulation groups

Simulation groups: controls which particle simulation groups will be assigned to the particles in the burst.

Export groups

Export groups: controls which particle export groups will be assigned to the particles in the burst.

Custom float

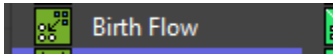
Value: the custom float value to assign to the particles in the burst.

Channel: the custom float data channel the value will be assigned to.

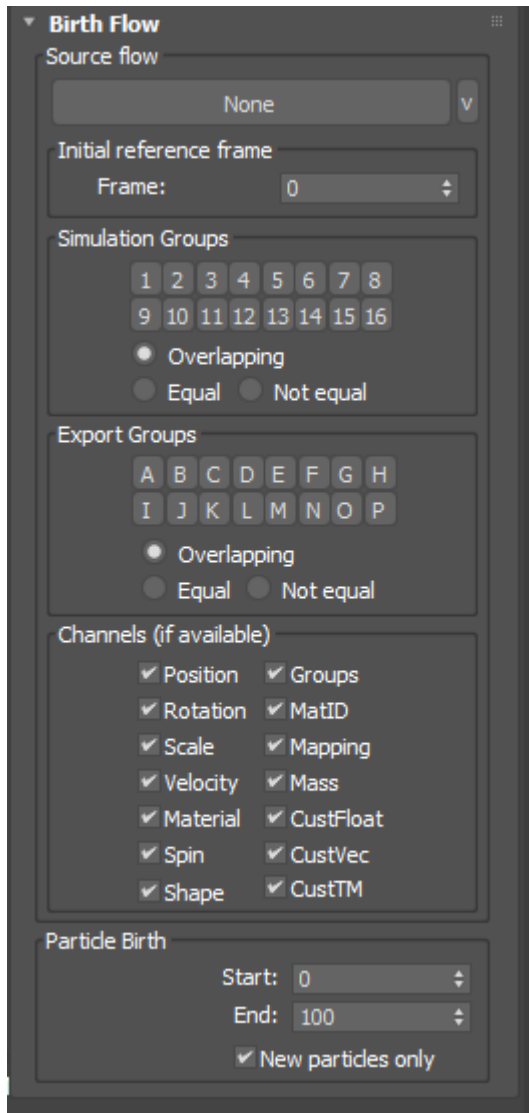
TIP

The Birth Burst operator is a more efficient alternative to creating multiple unique birth operators in order to birth distinct groups of particles.

Birth Flow operator



The Birth Flow operator allows you to birth new particles that are copies of particles from another flow.



Source flow

- **Flow object:** the **tyFlow** object whose particles will be copied.

Initial reference frame

Frame: the initial frame of reference that the input flow object will be evaluated at.

Simulation Groups

Controls which particle simulation groups will be read from the input flow object.

Channels (If available)

Controls which particle data channels to copy from the input flow object's particles.

NOTE: Birth Flow cannot import bindings/cloth/PhysX data/actor-dependencies/etc from other flows. Only data from the selected channels will carry over.

Particle birth

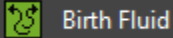
Start/End: controls the time range in which to birth new particles, copied from the input flow object.

New particles only: controls whether only new particles will be birthed each frame. A particle is considered new if its input ID has not been processed on a previous frame.

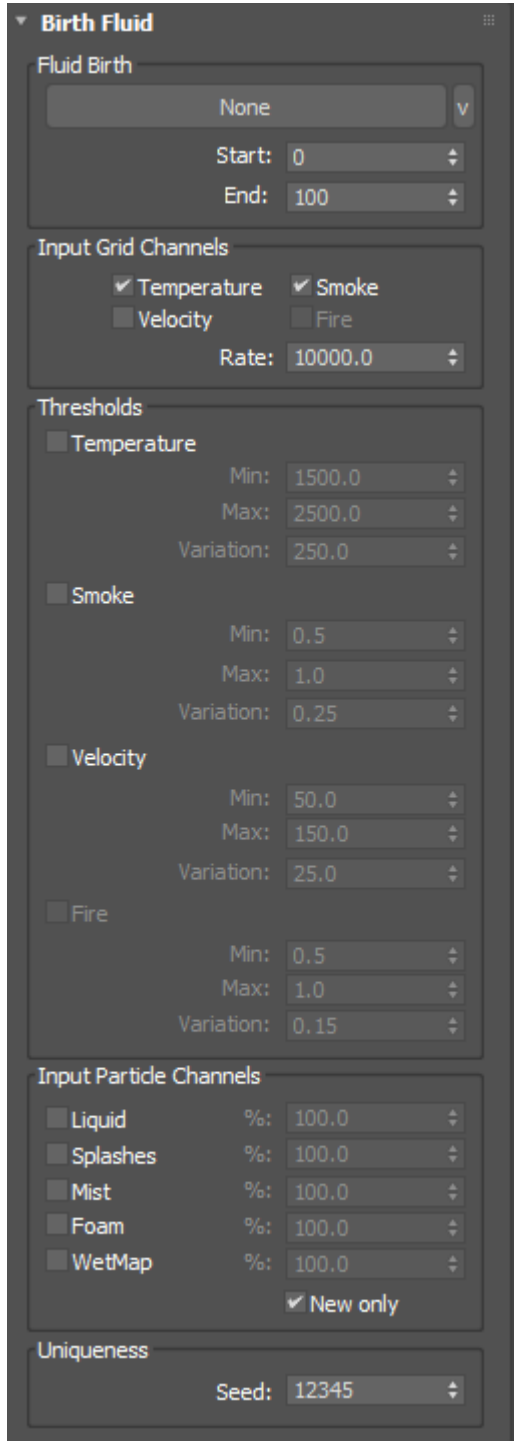
TIP

Birth flow operators are useful for optimizing the initial states of flows. For example: imagine a flow that computes a voronoi fracture on 1000 particle meshes at frame 0. Every time a property of that flow is changed, the flow's simulation will reset and the voronoi fracture will be recomputed in the process. To avoid having to wait for the fractures to initialize each time the flow property is changed, a secondary flow could be created that uses a birth flow operator in order to copy the fractured particles from the first. Each time the second flow's properties change, the voronoi fracture of the input flow **will not** have to be recomputed as well. And each time the first flow's settings change, the second flow will be automatically updated in the process. In this way, flows can be connected together in order to minimize recomputation costs of complex initial state parameters.

Birth Fluid operator



The Birth Fluid operator can be used to birth particles inside of a (Phoenix FD) fluid grid based on grid voxel and particle properties.



Fluid Birth

Fluid object: the input fluid grid object.

Start/End: controls the time range in which to birth new particles.

Input grid channels

Temperature: when enabled, grid cells with a temperature value above 300 will be able to spawn particles.

Smoke: when enabled, grid cells with a smoke value above 0 will be able to spawn particles.

Velocity: when enabled, grid cells with a velocity value above 0 will be able to spawn particles.

Fire: when enabled, grid cells with a fire value above 0 will be able to spawn particles (FumeFX only).

NOTE: The higher the temperature/smoke value per cell, the higher the probability that a particle will be spawned in the cell.

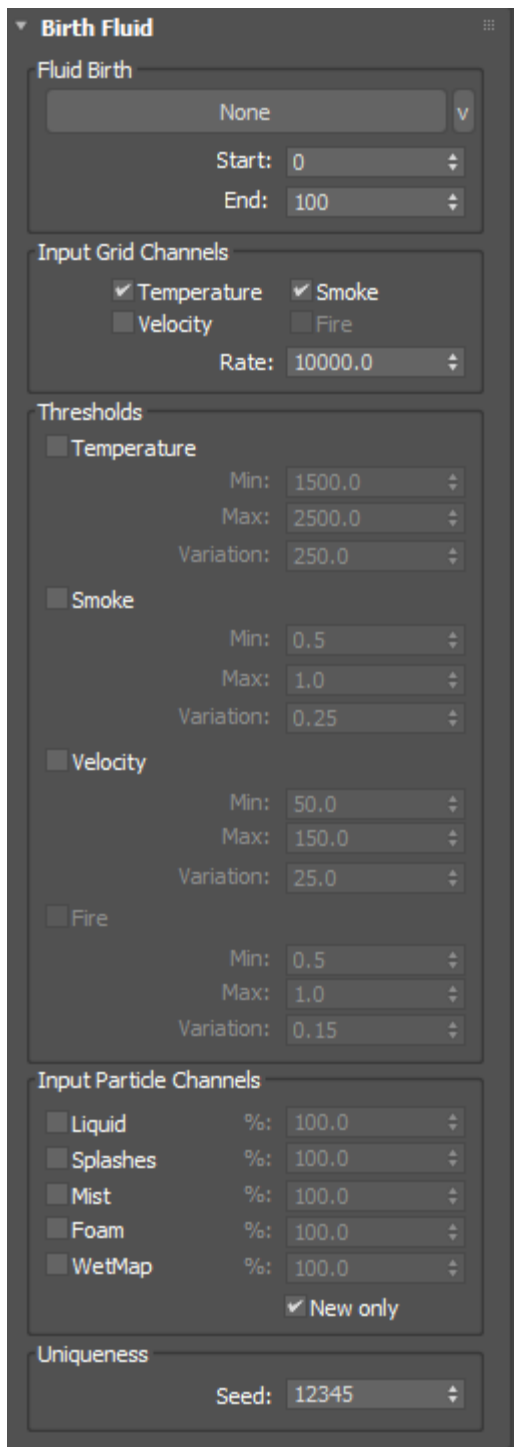
Rate: The theoretical rate of particles births that can happen per simulation step, given a per-cell probability of 1. The actual rate of particle births will be lower, and the birth probability per-cell is dependent on the total smoke and temperature values of each cell.

Thresholds

Enabling these channels will override the default cell spawn probabilities.

Temperature/Smoke/Velocity/Fire: the threshold override channels.

Min/Max/Variation: the range of values for each selected channel that will control how many particles are spawned in each cell.



Input Particle Channels

PhoenixFD grids can contain both voxels and particles. To birth particles that will be copies of PhoenixFD grid particles, enable the relevant particle channels.

Channels: enabling these channels will birth copies of particles contained within them, if such particles are available in the cache of the input object.

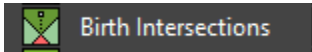
Channel percentages: controls the percentage of copied input particles to birth.

New only: controls whether only new particles will be birthed each frame. A particle is considered new if its input ID has not been processed on a previous frame.

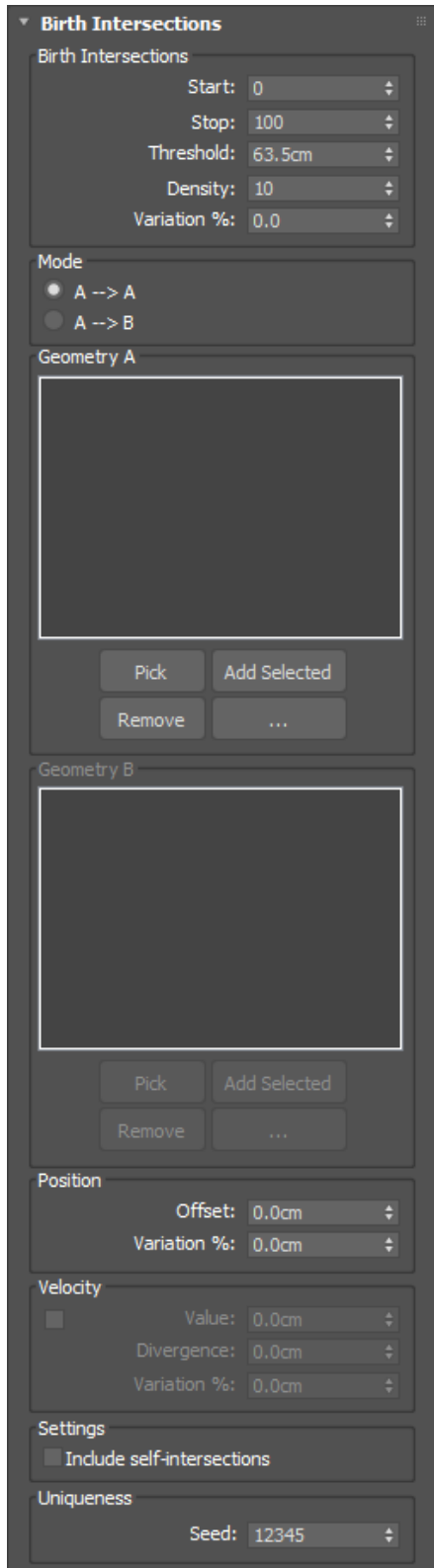
Uniqueness

Seed: the seed value for all varied parameters.

Birth Intersections operator



The Birth Intersections operator allows you to birth new particles at places where the geometry of input objects intersects.



Birth Intersections

Start/End: controls the time range in which to birth new particles at intersection points.

Threshold: affects the density of particles birthed on two intersecting faces by multiplying the density value by the ratio between this value and the length of the overlap between the intersecting faces.

Density: controls the base number of particles to birth over each intersection, prior to adjustments made relative to the threshold value.

Variation %: the per-particle percentage of variation to apply.

INFO:

The number of particles birthed on any given intersecting face can be estimated as:

(density) x (length of intersection overlap / threshold).

Mode

A > A: this mode will compute intersections between objects in Geometry List A.

A > B: this mode will compute intersections between objects in Geometry List A and Geometry List B. Intersections between objects in the same group (A > A or B > B) will not be computed.

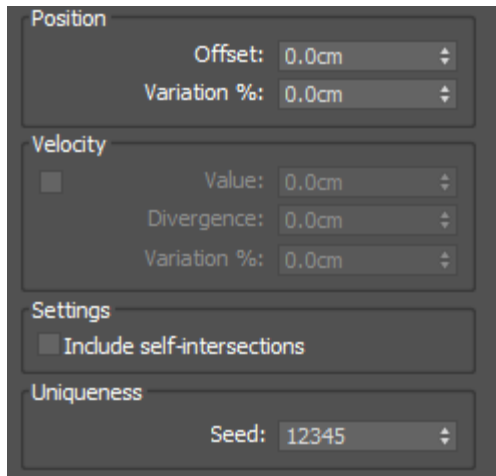
Position

Offset: controls the amount of offset along face normals to position the birthed particles.

Variation %: the per-particle percentage of variation to apply.

Velocity

Enable velocity: controls whether velocity along face normals is added to birthed particles.



Velocity value: the amount of velocity along face normals to add to birthed particles.

Divergence: the angle of divergence applied to velocity vectors.

Variation %: the per-particle percentage of variation to apply.

Settings

- **Include self-intersections:** controls whether self-intersections will be computed for objects.

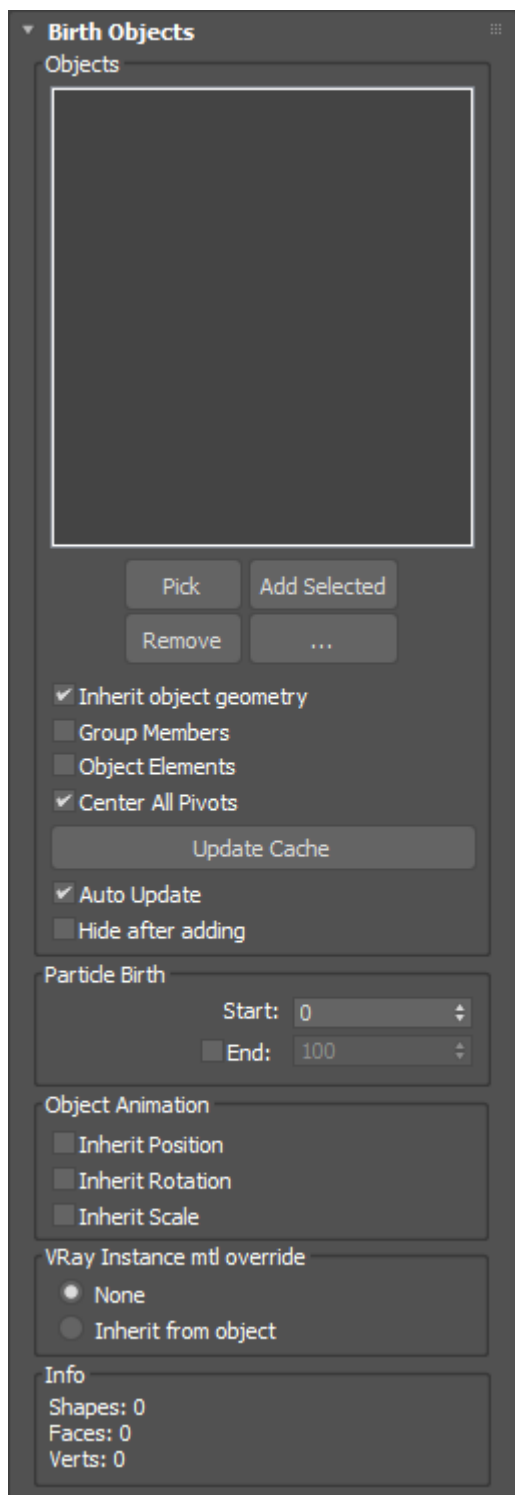
Uniqueness

- **Seed:** the seed value for all varied parameters.

Birth Objects operator

Birth Objects

The Birth Objects operator allows you to convert scene objects into particles.



Objects

Object list: the list of input objects which will be converted into particles.

Inherit object geometry: controls whether the meshes of input objects will be assigned to corresponding particles.

Note: When “inherit object geometry” is disabled, only object transforms will be transferred to birthed particles. This can provide a significant speedup if the input objects have high resolution meshes, but you do not need those meshes to be assigned to the new particles (for example, if you want to align particles to objects but don’t need those particles to inherit the meshes).

Group members: if an input object is a group head, enabling this setting will convert its constituent members into particles.

Object elements: if an input object mesh has multiple sub-elements, enabling this setting will split them into multiple particles.

Center all pivots: centers the pivots of extracted meshes.

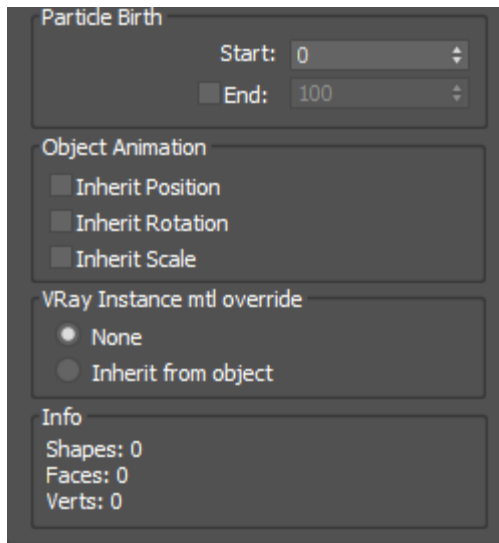
Update Cache: click this to manually update the internal cache which holds mesh data for all input objects.

Auto update: controls whether changes to input objects will automatically update the operator’s internal mesh cache.

Hide after adding: controls whether objects will be hidden in the scene after adding them to the listbox.

Note: The “hide after adding” setting is a sticky setting (remembered between operators and Max sessions) that will remain checked/unchecked depending on which state you leave it in last.

Birth Objects Operator Continued



Particle Birth

Start: the start frame at which particles will be birthed.

End enabled: controls whether particles should be birthed over a range of frames.

End: the end frame at which particles will be birthed.

Object Animation

Note: If particles leave the event which contains the Birth Objects operator, their animation will no longer be updated by the Birth Objects operator.

Inherit Position: particles will inherit the position changes of the scene object they reference.

Inherit Rotation: particles will inherit the rotation changes of the scene object they reference.

Inherit Scale: particles will inherit the scale changes of the scene object they reference.

VRay Instance mtl override

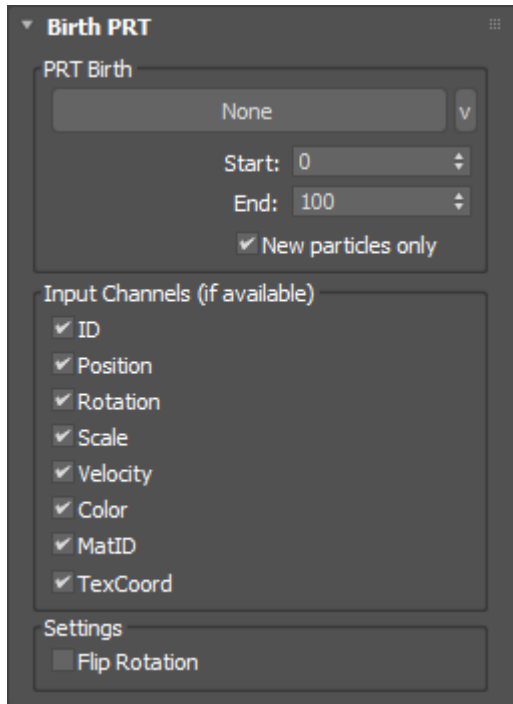
None: no material override will be assigned to render instances of the born particles.

Inherit from object: the material override for render instances of born particles will be taken from the scene object they are referencing.

Birth PRT operator



The Birth PRT operator allows you to birth new particles that are copies of particles from a PRTLoader object.



PRT birth

PRT object: the input object that contains PRT particles.

Start/End: controls the time range in which to birth new particles.

New particles only: controls whether only new particles will be birthed each frame. A particle is considered new if its input ID has not been processed on a previous frame, or its age is 0.

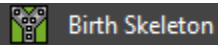
Input channels

Channels: controls which PRT data channels to copy into to birthed particles.

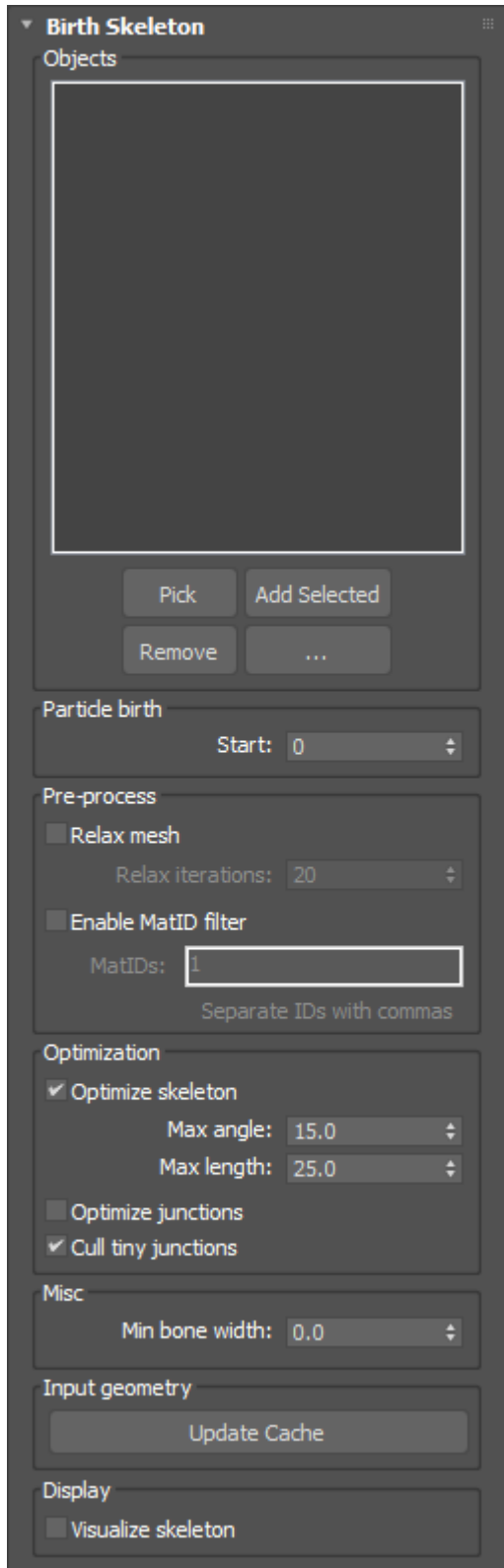
Settings

Flip rotation: flips the w-component of quaternions loaded from PRT data.

Birth Skeleton operator



The Birth Skeleton operator can be used to extract the curve skeleton from input meshes.



INFO:

The Birth Skeleton operator extracts the curve skeleton from input meshes, and converts the segments of bones within the extracted skeleton into particles. Using this method, you can easily rig complex meshes like trees and foliage which would otherwise be extremely difficult and time-consuming to manually rig.

TIP:

Within the context of this operator, a “segment” is a single line used to compose a bone. A “bone” is a curve composed of one or more “segments”. A “skeleton” is the set of all “bones” extracted from a mesh. Keep in mind that the skeleton extracted from a mesh is not a typical anatomical skeleton (ie, this operator is not meant to extract bones for the purpose of character rigging), but a curve skeleton (ie, the algorithm will attempt to convert tube-like sections of a mesh into segmented curves). For this reason, the operator is ideal for extracting bones from tube-like meshes (that’s why it works great for plants, foliage, tentacles, etc), but it doesn’t work very well for meshes that do not have many long, protruding parts.

Objects

Object list: the list of input objects whose skeleton will be extracted.

Particle birth

Start: controls the frame at which to birth new particles.

Pre-process

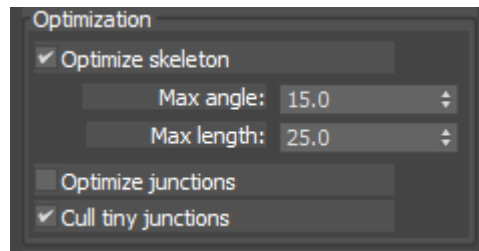
Relax mesh: applies laplacian relaxation to the input mesh, in order to smooth out surface details prior to skeletonization.

Relax iterations: the number of relax iterations to apply to the input mesh. The higher the iterations, the smoother the mesh will be.

Enable MatID filter: when enabled, only faces matching the listed material ID values will be skeletonized.

MatID: the list of face material IDs to skeletonize.

Birth Skeleton Operator Continued



TIP:

Input meshes with a lot of small, bumpy surface details can cause artifacts to appear in the resulting skeleton (junctions with unnecessary bones pointing in random directions, bones not properly following the overall surface curvature, etc). Performing pre-process relaxation (or manually relaxing the geometry of the mesh yourself) can help to improve the result of the skeletonization process. An ideal mesh for skeletonization is one that is perfectly smooth and composed only of long, tube-like structures.

TIP:

Using the material ID filter is an easy way to exclude parts of a mesh (like leaves, small twigs, etc) from skeletonization.

Optimization

Optimize skeleton: controls whether or not the raw extracted skeleton will be optimized in various ways.

Note: The max angle and max length settings control how individual bones will be optimized, by merging/reducing their segments.

Max angle: connected segments whose angle is below this threshold will be merged.

Max length: segments below the max angle threshold will only be merged if their combined length is below this value.

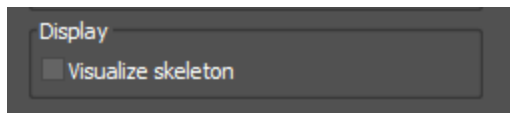
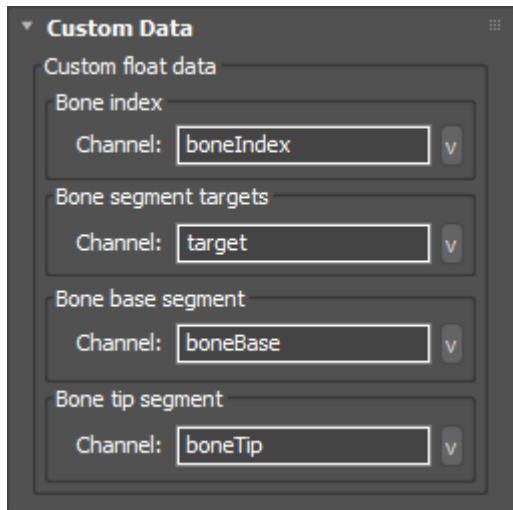
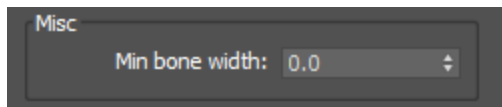
Optimize junctions: in skeletons with clear junctions (places where the endpoint of a bone touches the midpoint segment of another bone), segments will be merged at junction points.

Note: Turning “optimize junctions” on may cause endpoints of bones which touch midpoint segments of other bones to lose contact with those midpoint segments, if those midpoint segments are optimized away. Keeping this setting off will maximize junction contact in skeletons extracted from meshes with clean, contiguous topology.

Cull tiny junctions: single-segment bones generated at endpoint junctions of multi-segment bones will be culled.

Note: The skeletonization algorithm has a tendency to create junctions with multiple single-segment bones pointing outwards at seemingly-random angles, in meshes that aren't composed of perfectly smooth tube-like structures. This setting uses a fairly robust heuristic to minimize the creation of those bones.

Birth Skeleton Operator Continued



Misc

Min bone width: specified the minimum width of generated particles.

Custom float data

Bone index

Stores the bone index of each segment into a channel within the corresponding particle.

Channel: the custom data float channel where the bone index value will be stored.

Bone segment targets

Bone segments each have two endpoints (the start and end of each segment line), and bones are composed of multiple segments ordered from base to tip. Thus, the target of each segment is the segment before it (excluding the first segment in a bone, which has no target). Thus, the bone segment target of a corresponding particle will be the particle ID generated before it, as each bone is processed.

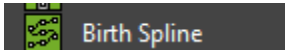
Channel: the custom data float channel where the bone segment target value will be stored.

Display

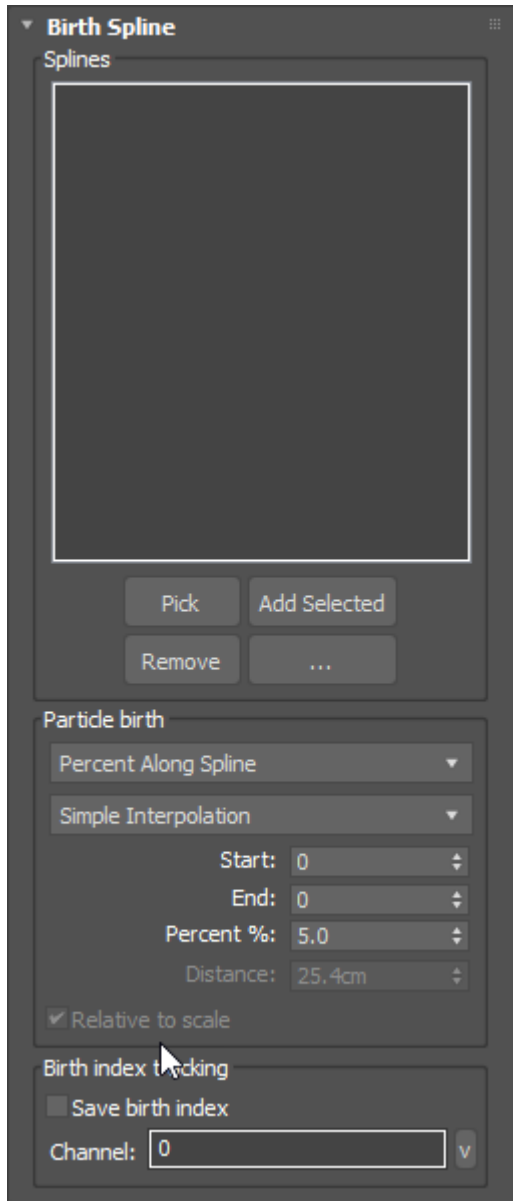
Visualize skeleton: draws the raw bone segment lines in the viewport, on the frame they are generated.

Note: Each bone is given a random color, and the brightness of the color applied to each bone segment increases from base to tip. Thus, you can see the directionality of each bone in the viewport by looking at the direction of the color gradient for each bone when visualization is enabled. Bone directionality is not controllable, and is determined internally using an algorithm that examines bone radius, proximity, and direction to nearby bones.

Birth Spline operator



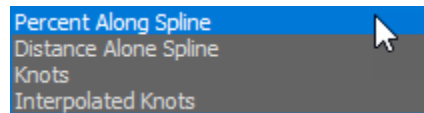
The Birth Spline operator can be used to birth new particles on the curves of splines.



Splines

- **Spline list:** the list of input splines whose curves will be used to birth particles.

Particle birth

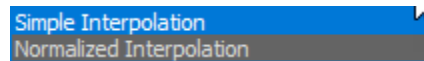


Percent Along Spline: particles will be birthed at intervals along splines based on a percentage of their total length.

Distance Along Spline: particles will be birthed at intervals along splines based on a distance along their total length.

Knots: particles will be birthed on spline knots.

Interpolated Knots: particles will be birthed on implicit spline knots, whose interpolated locations are based on the spline's step and optimization settings.



Simple Interpolation: percents and distances along the splines will be relative to sub segment lengths and knot spacing.

Normalized Interpolation: percents and distances along the splines will be relative to the normalized spline length, ignoring segments lengths and knot spacing.

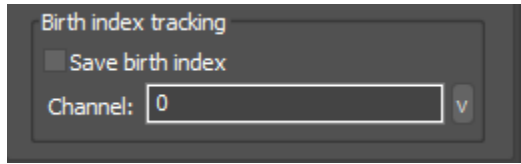
Start/End: controls the time range in which to birth new particles.

Percent %: the percent value used in “percent along spline” mode.

Distance: the distance value used in “distance along spline” mode.

Relative to scale: the distance value will be normalized against the magnitude of the spline object's scale vector, so that generated points will be the same distance apart in world space.

Birth Spline Operator Continued



Birth index tracking

Save birth index: controls whether the 0-based index of the spline in the input object list that a particle was birthed on is saved to a the particle's custom data channel.

Note: The birth index value will be additionally offset by the number of subsplines the input objects have. So, each particle will be given a unique birth index relative to both the input object and its corresponding subspline count. So if two objects are in the list and each object has 1 subsplines, the particle birthed on the first subspline will have an index of 0, and the particle birthed on the last subspline will have an index of 1. If there are two objects in the list, and the first object has 2 subsplines and the second object has 5 subsplines, the particle birthed on the first subspline will have an index of 0, and the particle birthed on the last subspline will have an index of 6.

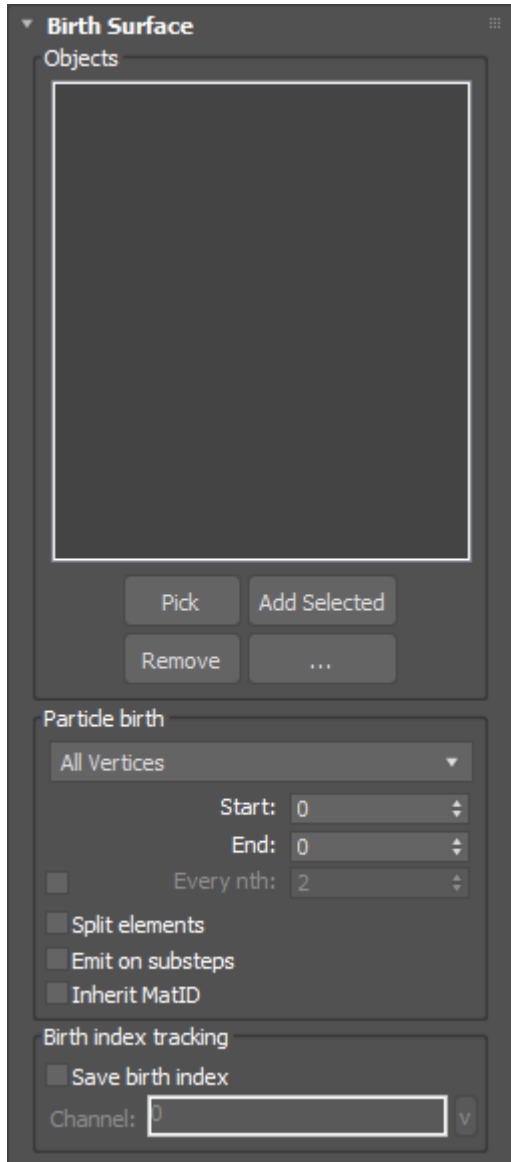
Channel: sets the data channel the birth index will be saved to

Birth Surface operator



Birth Surface

The Birth Surface operator can be used to birth new particles on specific input mesh features, such as vertices, face centers, etc.



Note: Instead of scattering particles randomly on surface features, particles will be sequentially birthed on surface features in the order that the features are indexed. For example, when birthing particles on surface vertices, particle 1 will birth at the location of vertex 1, particle 2 will birth at the location of vertex 2, etc.

Objects

Object list: the list of input objects whose surfaces will be used to birth particles.

Particle birth

Surface feature type: the surface feature on which particles will be birthed.

Start/End: controls the time range in which to birth new particles.

Enable every nth: when enabled, particles will be birthed at regular frame intervals, instead of at every frame.

Nth value: the interval value at which to birth particles.

Split elements: controls whether the internal spawn parent value of particles will be incremented depending on which element of a surface the particle is birthed on.

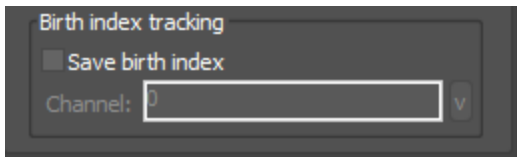
Emit on substeps: controls whether particles will be birthed on sub steps of the simulation, or only on whole frames.

Inherit matID: controls whether the material ID property of the underlying surface will be copied to the appropriate particle data channel.

INFO:

When a particle is birthed on a surface, its internal spawn parent value is set to the index of that surface in the object list, relative to the starting birthID of particles in the event. This means that all particles on the same surface are considered siblings of each other. Their sibling relationship can have an effect on whether they will consider each other binding candidates, among other things.

Sometimes you may not want all particles birthed on surface features to be siblings with each other. For example, if your input object is a shape with multiple spline sub-elements, you would not want particles birthed on separate sub-elements to be siblings of each other. By enabling “split elements”, the spawn parent value of birthed particles will not only be incremented for each object in the input list, but also for each sub element of each object. This will create proper sibling relationships between particles birthed on sub-elements of input surfaces, and make it easier to do things like convert input sub-element splines into separate constraint networks.



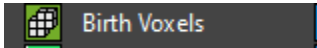
Birth index tracking

Save birth index: controls whether the 0-based index of the surface in the input object list that a particle was birthed on is saved to a the particle’s custom data channel

Note: The birth index value will be affected by whether or not “split elements” is enabled.

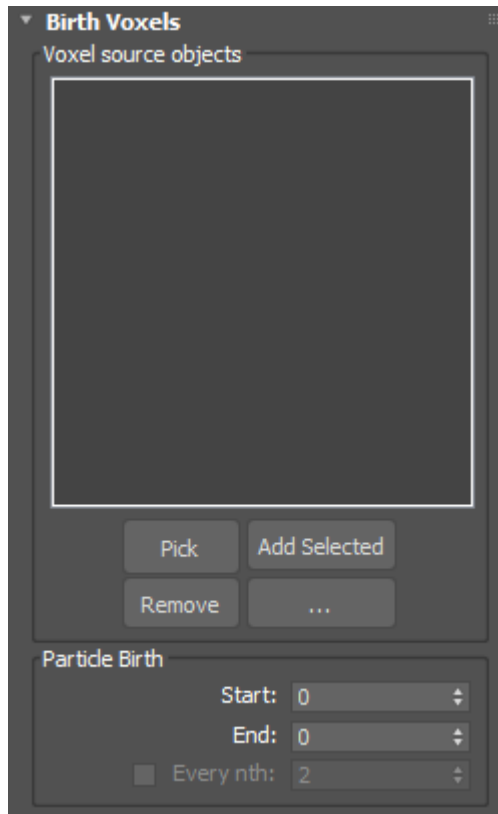
Channel: sets the data channel the birth index will be saved to.

Birth Voxels operator

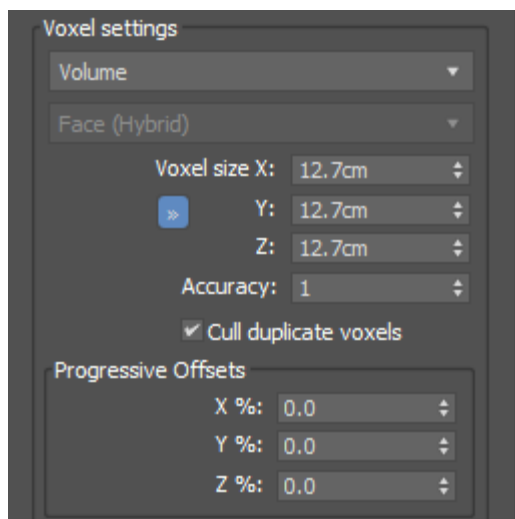


The Birth Voxels operator allows you to birth particles inside the volume or surface of scene objects, at uniform spatial intervals.

Top Part of Rollout



Middle Part of Rollout



Voxel source objects

Object list: the list of input objects whose volumes will be used to birth particles.

Particle birth

Start/End: controls the time range in which to birth new particles.

Enable every nth: when enabled, particles will be birthed at regular frame intervals, instead of at every frame.

Nth value: the interval value at which to birth particles.

Voxel settings

Mode: controls where particles will be birthed, relative to the surface or volume of a particular input object.

Sample type: controls the quality of samples used to determine surface proximity, when birthing particles in "Surface" mode.

Voxel size X/Y/Z: the size of individual voxels, in which particles will be birthed.

Accuracy: controls the accuracy of the raycaster used to compute information about whether a particle is inside or outside of an object's volume. Increase this value for meshes that are self-intersecting or have holes in them.

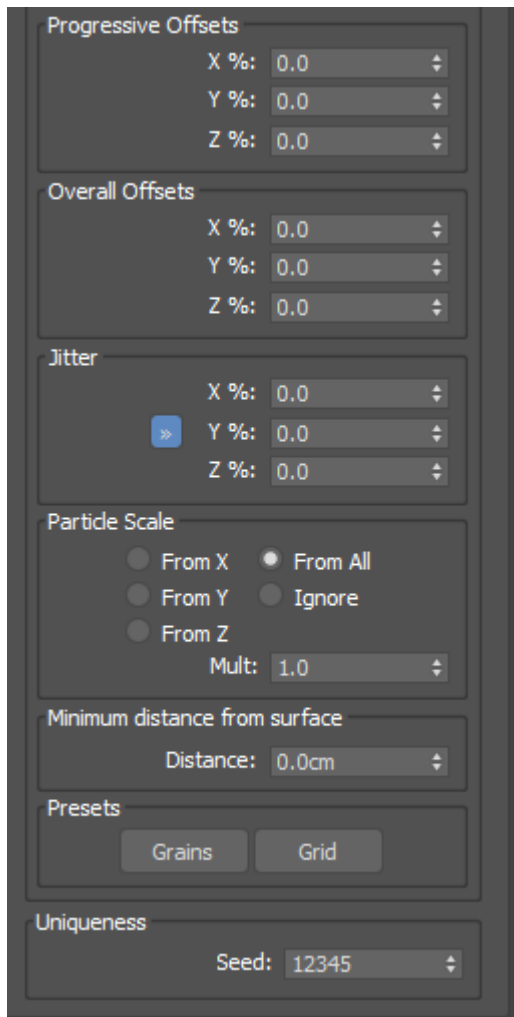
INFO:

If an input object's surface is not closed, or is self-intersecting, increasing the accuracy value can improve results and reduce artifacts.

Cull duplicate voxels: when voxelizing multiple input objects, enabling this setting will delete any overlapping particles generated between the objects.

Birth Voxels Operator Continued

Bottom Part of Rollout



Progressive offsets

X/Y/Z %: the percent of progressive position offset applied to each particle, relative to the overall size of the voxels.

Overall offsets

X/Y/Z %: the percent of overall position offset applied to each particle, relative to the overall size of the voxels.

Jitter

X/Y/Z %: the percent of overall position jitter applied to each particle, relative to the overall size of the voxels.

Particle Scale

Scale type: controls which size value particle will derive their scale from.

Mult: a global scale multiplier applied to all particles.

Minimum distance from surface

Distance: particles within this distance from the surface will be culled.

Presets

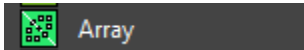
Grains: choosing this preset will apply a progressive offset to particles, such that their positions in space will form a tightly-knit overlapping grid.

Grid: choosing this preset will remove all offsets from particles, such that their positions in space will form a uniform, aligned grid.

Uniqueness

Seed: the seed value for all varied parameters.

Array operator



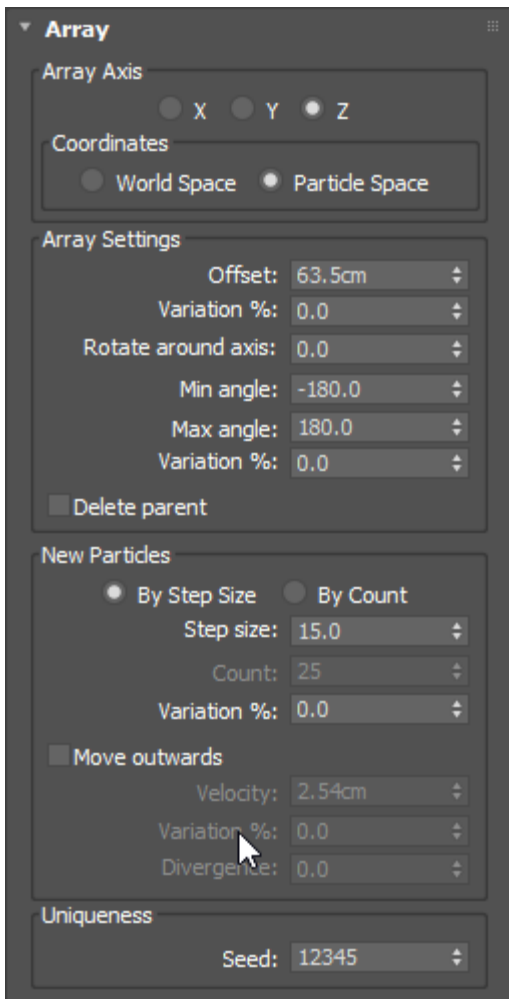
The Array operator allows you to spawn particles in circular patterns.



No help on this rollout at this time



No help on the rollout at this time



Array Axis

X/Y/Z: controls the axis around which particles will be spawned.

Coordinates

World Space: the selected axis will be relative to world-space.

Particle Space: the selected axis will be relative to the current particle's transform.

Array Settings

Offset: the distance between the current particle and spawned particles.

Variation %: the per-particle percentage of variation to apply.

Rotate around axis: controls how much rotational offset to apply to all spawned particles, around the array axis.

Min/Max angle: controls the size of the circular arc around the array axis that particles will be spawned within.

Variation %: the per-particle percentage of variation to apply.

Delete parent: when enabled, the parent particle of the new array particles will be deleted.

New Particles

By step size: controls whether particles will be spawned at steps (in degrees) around the array arc.

By count: controls whether a static number of particles will be spawned, regardless of the size of the array arc.

Array Operator Continued

The screenshot shows a control panel for the Array Operator. It features a 'Delete parent' checkbox at the top. Below it is the 'New Particles' section, which includes two radio buttons: 'By Step Size' (selected) and 'By Count'. Under 'By Step Size', there are three spinners: 'Step size' (15.0), 'Count' (25), and 'Variation %' (0.0). Below this is the 'Move outwards' section, which is currently disabled (checkbox is unchecked). It contains three spinners: 'Velocity' (2.54cm), 'Variation %' (0.0), and 'Divergence' (0.0). At the bottom is the 'Uniqueness' section, which has a 'Seed' spinner set to 12345.

Step size: the spawn step size, in degrees.

Count: the spawn count.

Variation %: the per-particle percentage of variation to apply.

Move outwards: when enabled, spawned particle velocities will be affected such that they move outward from the current particle.

Velocity: the scale of the outward velocity vector to apply to spawned particles.

Variation %: the per-particle percentage of variation to apply.

Divergence: controls the degrees of random divergence to apply to outward velocity vectors

Uniqueness

Seed: the seed value for all varied parameters.

Branch operator



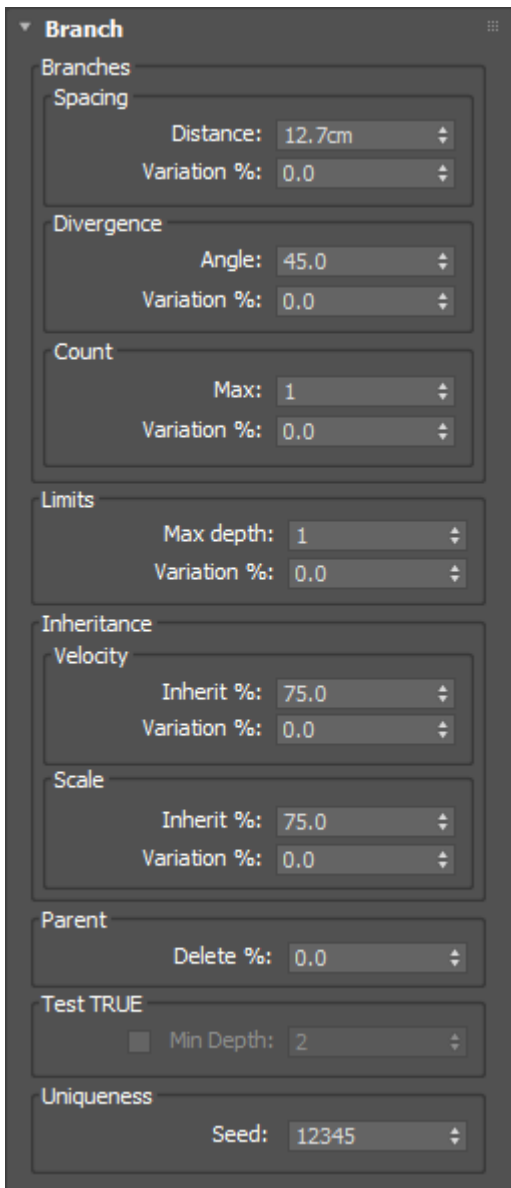
The Branch operator spawns child particles which travel at predictably-divergent angles relative to parent particles. It can be used as the basis for effects like growing frost, lightning, etc.



No help on this rollout at this time



No help on the rollout at this time



Branches

Spacing

Distance: the minimum distance a parent particle must travel before spawning a child.

Variation %: the per-particle percentage of variation to apply.

Divergence

Angle: the angle of divergence applied to spawned particles' velocity trajectory.

Variation %: the per-particle percentage of variation to apply.

Count

Max: the maximum number of particles to spawn at each branch event.

Variation %: the per-particle percentage of variation to apply.

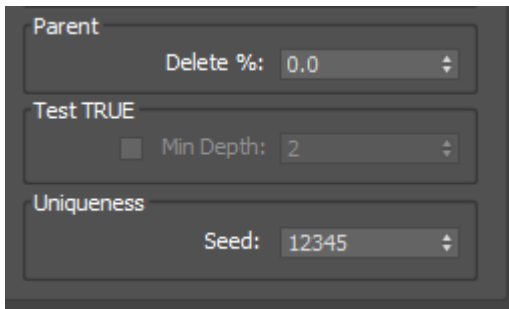
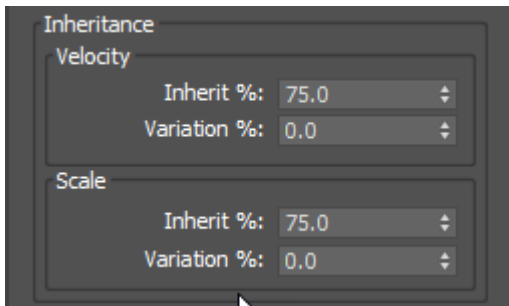
Limits

Max depth: the maximum number of recursive branches that may be spawned.

Variation %: the per-particle percentage of variation to apply.

Note: Increasing max depth increases the number of successive recursive steps the branching algorithm can take. For example, a max depth of 2 means that particles which enter the event may spawn children, and those children may spawn children of their own, but that third group of children (the grand-children of the original particles) may not spawn further children. Increasing

Branch Operator Continued



the max depth will exponentially increase the total number of particles that will be spawned over time.

Inheritance

Velocity

Inherit %: the amount of velocity child particles will inherit from parent particles.

Variation %: the per-particle percentage of variation to apply.

Scale

Inherit %: the amount of scale child particles will inherit from parent particles.

Variation %: the per-particle percentage of variation to apply

Parent

No Help at This Time

Test TRUE

Min depth enable: controls whether child particles that reach a certain depth will test TRUE for output.

Min depth value: sets the minimum depth value required for child particles to satisfy the condition.

Uniqueness

Seed: the seed value for all varied parameters.

