



Particle Flow Tools[®]: Box#3 Pro

version 1.5

Reference Documentation



Copyright © 2009 Orbaz Technologies, Inc. All rights reserved.

License Agreement

Orbaz Technologies, Inc. Computer End User License Agreement (EULA) for Particle Flow Tools® Box #3 Pro

IMPORTANT -- PLEASE READ CAREFULLY: This Orbaz Technologies, Inc. ("Orbaz") End User License Agreement ("EULA") is a legal agreement between you (either an individual or an entity) and Orbaz for the software product ("Software") mentioned above. By installing, copying or otherwise using the Software, you agree to be bound by the terms of this EULA. If you do not agree to the terms of this License Agreement, do not install or use the Software.

=====

NOTE: THIS PRODUCT MUST BE REGISTERED BEFORE YOU CAN USE IT.

=====

1. General

This is a license agreement and NOT an agreement for sale. Under this contract Orbaz grants to you a non-exclusive license to use the Software, which is software and documentation. The Software itself as well as the copy of the Software or any other copy you are authorized to make under this contract remain the property of Orbaz at all times.

2. Use of the Software

- Orbaz grants you a nonexclusive, nontransferable license to use the Software and its manual and other accompanying printed material and "online" or electronic documentation with equipment owned by you or under your control, according to the terms and conditions of this Agreement. This Agreement permits a single user to install and use the Software on only one computer at one location at any one time.
- If the Software is identified as a demonstration, evaluation, or NFR version, you may use it only for the purpose of commercial evaluation and demonstration. Such licenses are generated for a specific fixed time period. After a NFR license has been expired, all related documentation and data must be destroyed or sent back to Orbaz or the dealer who handled the NFR version. You may not use it for commercial, professional, or for-profit purposes.

3. Multiple Use and Network Operation

If this Software is a Network Version, you may use it only over an internal local area network environment with the Orbaz Floating License Tool, and you may install and operate the Software on a single server computer in a single location which may be accessed by other computers, or on an individual computer, as a multiple-user installation with either:

- The maximum number of concurrent users being one (1), so that multiple individuals may access or use the Software, but that only one person at a time may do so, or
- The maximum number of concurrent users being more than one (1), in which case you must purchase single seat licenses for each additional concurrent user. The use of software or any device that reduces the number of computers/devices which access the Orbaz Floating License Tool when used in a Server configuration may interfere or damage the licensing tool or prevent the Software from running properly. In no case will such a device "reduce" or prevent you from buying the number of single seat licenses required.

4. Transfer

- You may not rent, lease, sublicense or lend the Software or documentation. You may, however, transfer all your rights to use the Software to another person or legal entity provided that you transfer this agreement, the Software, including all copies, updates or prior versions. You must inform Orbaz Technologies, Inc. in writing about a license transfer and the new user has to sign and accept this license agreement.
- In case of such a transfer the license of the former user expires and all remaining data covered by this license must be deleted/destroyed.

5. You May Not:

- Copy or use the Software or Documentation except as permitted by this Agreement.
- Reverse engineer, decompile, or disassemble the Software except to the extent permitted by law where this is indispensable to obtain the information necessary to achieve interoperability of an independently created program with the Software or with another program and such information is not readily available from Orbaz or elsewhere.
- Install or use the Software on the Internet or over a wide area network, including, without limitation, use in connection with a Web based render farm or similar service.
- Remove, alter, or obscure any proprietary notices, labels, or marks from the Software or documentation.
- Utilize any equipment, device, software, or other means designed to circumvent or remove any form of copy protection used by Orbaz in connection with the Software, or use the Software together with any authorization code, serial number, or other copy protection device not supplied by Orbaz directly or through an authorized reseller.

6. Limited Warranty

SPECIAL EFFECTS AND RENDERING PLUGINS ARE TOOLS INTENDED TO BE USED BY TRAINED PROFESSIONALS ONLY. ORBAZ WARRANTS THAT THE SOFTWARE WILL PERFORM IN ACCORDANCE WITH THE DOCUMENTATION. ORBAZ CAN NOT WARRANT THAT THE SOFTWARE WILL WORK TOGETHER WITH OTHER SOFTWARE AND PLUG-INS FROM OTHER 3RD PARTY DEVELOPERS, BECAUSE OF THE COMPLEXITY OF SUCH INTERACTIONS BETWEEN DIFFERENT OPERATING SYSTEMS OR SOFTWARE PACKAGES. THE USER MAY HOWEVER IMMEDIATELY REPORT SUCH INCOMPATIBILITIES FOR FURTHER INSPECTION BY ORBAZ. SUCH A REPORT HAS TO BE DONE IN WRITING.

7. Warranty Disclaimer

THE INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. ORBAZ DISCLAIMS ALL WARRANTIES, EITHER EXPRESSED OR IMPLIED, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL ORBAZ BE LIABLE FOR ANY DAMAGES WHATSOEVER INCLUDING DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, LOSS OF BUSINESS PROFITS OR SPECIAL DAMAGES, EVEN IF ORBAZ HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES SO THE FOREGOING LIMITATION MAY NOT APPLY.

8. Confidentiality

All Licensing information, including license files, descriptions of code activation and written instructions of any kind created by Orbaz are only intended for the licensed user of the Software and no one else. Such information may not be spread or distributed in any form to other users.

9. Registration Data

For the purpose of customer registration and control of proper use of the programs Orbaz will store personal data of the users in accordance with the German law on Personal Privacy and Data Protection. This data may only be used for the above-mentioned purposes and will not be accessible to third parties.

10. Miscellaneous

If any provision of this Agreement is found to be invalid or otherwise unenforceable, the further conditions of this Agreement will remain fully effective and the parties will be bound by obligations which approximate, as closely as possible, the effect of the provision found invalid or unenforceable, without being themselves invalid or unenforceable.

BY INSTALLING AND AUTHORIZING THE SOFTWARE, YOU HEREBY AGREE TO THE TERMS AND CONDITIONS PRESENTED ABOVE.

Particle Flow Tools® is a registered trademark of Orbaz Technologies, Inc.

Table Of Contents

| | |
|--|----|
| Reference Documentation..... | 1 |
| License Agreement | 2 |
| Table Of Contents | 4 |
| Introduction..... | 8 |
| What's New in the 1.5 Upgrade..... | 9 |
| Quick Start Tutorial | 10 |
| Part 1: Create a Data Operator | 10 |
| Use the Icon to Control Particle Motion..... | 15 |
| Replace the Icon with a Reference Object..... | 17 |
| Part 2: Save the Data Operator as a Standard Operator | 19 |
| Data View | 24 |
| Wiring Suboperators | 25 |
| Dynamic Suboperator Names | 25 |
| Suboperator Controls | 25 |
| Data Inputs | 26 |
| Filter Input | 26 |
| Suboperator Blocks..... | 27 |
| Grouping Suboperators | 29 |
| Additional Controls..... | 30 |
| Menus..... | 32 |
| Edit menu | 32 |
| Select menu..... | 32 |
| Group menu | 32 |
| Display menu | 32 |
| Options menu | 32 |
| Right-Click menu..... | 33 |
| Data View Hotkeys..... | 34 |
| Action..... | 34 |
| Default Shortcut..... | 34 |
| Data Types | 35 |
| Pair and Complex Data Types | 35 |
| Object suboperator – Point Position option | 35 |
| Geometry suboperator – Closest Point option | 36 |
| Geometry suboperator – Collision Point option | 36 |
| Geometry suboperator – Face Area and Face Selection options | 36 |
| Geometry suboperator – Point Color, Point Color Gradient, Point Self-Illumination, Point Mapping, Point Mapping Gradient, Point Material Index, Point Normal, Point Opacity, Point Position, Point Soft Selection, Point Speed options | 36 |
| Geometry suboperator – Random Surface Point option | 36 |
| Geometry suboperator – Random Volume Point option..... | 36 |
| Equal Data Type | 37 |
| Compound Index..... | 37 |
| Priority and Execution Order | 38 |
| History-Dependent..... | 40 |

| | |
|---|----|
| Data Icon/Operator/Icon Test/Test | 41 |
| Interface | 41 |
| Operator Activity group..... | 42 |
| Operator Icon group..... | 42 |
| Expose Parameters Dialog | 44 |
| Interface | 44 |
| Save Data Operator Preset Dialog | 47 |
| Procedure | 47 |
| To delete and organize saved operators and presets: | 47 |
| Interface | 47 |
| Preset/Operator Icon group..... | 48 |
| Amount Change Suboperator..... | 49 |
| Procedure | 49 |
| Interface | 52 |
| Priority And Execution Order group..... | 53 |
| Record Spawning To Data group..... | 53 |
| Color Suboperator..... | 55 |
| Interface | 55 |
| Condition Suboperator | 57 |
| Interface | 57 |
| Convert Suboperator | 60 |
| Discretizator Suboperator | 61 |
| Interface | 62 |
| Function Suboperator..... | 63 |
| Interface | 63 |
| First Operand group | 63 |
| Second Operand group..... | 64 |
| Result group..... | 65 |
| Geometry Suboperator | 68 |
| Interface | 68 |
| Objects group..... | 69 |
| Separate Indexing For group..... | 69 |
| Data Wiring group | 70 |
| Uniqueness group..... | 71 |
| Icon Suboperator..... | 72 |
| Interface | 72 |
| Subframe Sampling group | 73 |
| Data Wiring group | 73 |
| Uniqueness group..... | 73 |
| Input Custom Suboperator | 74 |
| Interface | 74 |
| Priority System Indexing group..... | 74 |
| Input Proxy Suboperator | 75 |
| Interface | 76 |
| Proxy System Indexing group..... | 77 |
| Input Standard Suboperator | 81 |

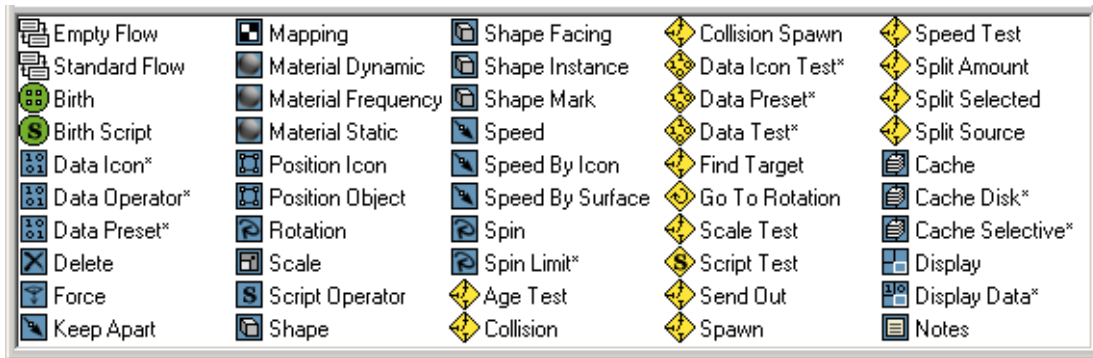
| | |
|---|-----|
| Interface | 82 |
| Notes Suboperator..... | 83 |
| Object Suboperator | 84 |
| Interface | 84 |
| Objects group..... | 84 |
| Data Wiring group | 85 |
| Object Parameter Suboperator | 86 |
| Interface | 86 |
| Output Custom Suboperator | 88 |
| Interface | 88 |
| Priority And Execution Order group..... | 88 |
| Output New Suboperator | 89 |
| Interface | 89 |
| Priority And Execution Order group..... | 89 |
| Output Standard Suboperator..... | 90 |
| Interface | 90 |
| Priority And Execution Order group..... | 92 |
| Output Test Suboperator | 93 |
| Interface | 93 |
| Priority And Execution Order group..... | 93 |
| Parameter Suboperator..... | 94 |
| Interface | 94 |
| Particles Suboperator | 96 |
| Interface | 96 |
| Aggregated Property group..... | 97 |
| Vicinity Radii and FOV group..... | 98 |
| Pipe Suboperator..... | 100 |
| Interface | 100 |
| Pipe Conditions..... | 101 |
| Random Suboperator | 102 |
| Interface | 121 |
| Distribution group..... | 121 |
| Scalar Suboperator | 123 |
| Interface | 123 |
| Uniqueness group..... | 125 |
| Select Object Suboperator..... | 126 |
| Interface | 126 |
| Shape Control Suboperator | 128 |
| Interface | 128 |
| Switch Suboperator..... | 130 |
| Interface | 130 |
| Vector Suboperator | 132 |
| Variation group..... | 133 |
| Uniqueness group..... | 134 |
| Cache Disk Operator..... | 135 |
| Interface | 136 |

| | |
|--|-----|
| Cache Files group | 138 |
| Update And Reset group..... | 138 |
| Cache Selective Operator..... | 141 |
| Interface | 142 |
| Update And Reset group..... | 144 |
| Memory Used (K) group..... | 145 |
| Display Data Operator | 147 |
| Interface | 147 |
| Show Data Channel group | 148 |
| Updated Particle Flow Operators..... | 150 |
| Shape Instance | 150 |
| Age Test..... | 150 |
| Material Inheritance | 151 |
| Interface | 155 |
| Update Materials group..... | 156 |
| Additional Parameter: Material/Birth Group/Shape Instance operators..... | 156 |
| Customizing Particle Flow Tools: Box#3 Pro | 157 |

Introduction

Thank you for your interest in *Particle Flow Tools® Box#3 Pro* plugin. The plugin is created to extend the capabilities of the build-in Particle Flow system in Autodesk® 3ds Max®.

The Particle Flow Tools operators appear in the Particle View depot along with the standard Particle Flow operators. The name of each PF Tools operator is followed by an asterisk (*) to differentiate it from standard tools.



Particle Flow Tools: Box#3 Pro includes Cache Disk, Cache Selective, Display Data, six data operators and 27 data suboperators. Also included are several custom operators created with *PFTools: Box#3 Pro*, including Blur Wind, RandomWalk, and Spin Limit. You can use these as is, or you can edit them in Data View by adding a Data operator or test and then using Load Preset to load one.

Cache Disk and **Cache Selective** operators record particle states for faster feedback. Some particle properties can be excluded from caching, and extra operators can be called to restore these data. **Cache Disk** operator stores particle data in a disk file on a file per frame basis.

Display Data operator shows numerical data created by data operators.

Data Operator and **Data Icon** operators modify particle channel data by using sub-operators wired in Particle Data View. **Data Icon** operator has a visible icon in viewports.

Data Preset – as an operator and as a test – loads predefined data operators. After you add the Data Preset to Particle View, a list of available presets appears automatically; choose one from the list and then click OK.

Data Test and **Data Icon Test** modify and check particle channel data by using sub-operators wired in Particle Data View. **Data Icon Test** has a visible icon in viewports.

Particle Flow Tools are under constant development. If you find areas that need improvement within the software please do not hesitate to contact us. We would love to hear from you and the easiest way to contact us is by email.

What's New in the 1.5 Upgrade

This package includes the version 1.5 upgrade for *Particle Flow Tools Box#3 Pro*, which provides four new suboperators and the Material Inheritance feature. Following is a list of new features in this upgrade:

- **Object Parameter** – Gets a single parameter value from any object in the scene, including modifier settings and Particle Flow operators and tests.
- **Shape Control** – Assigns shapes to particles, with direct control of the shape parameters for each particle.
- **Discretizator** – Makes float or integer values more discrete, reducing a wide range of values to a more manageable subset by rounding up or down. This can provide a marked improvement in the speed of handling large particle systems.
- **Color** – Generates vector-type data, specifically for use in vertex-color mapping for particles.
- **Material Inheritance** – Improves the handling of the materials in a Particle Flow system via tools that simplify material assignment and inheritance.
- **Customization** – You can modify the INI file or use built-in tools to change default settings for any operator or suboperator.

Email: support@orbaz.com

Particle Flow Tools® is a registered trademark of Orbaz Technologies, Inc.

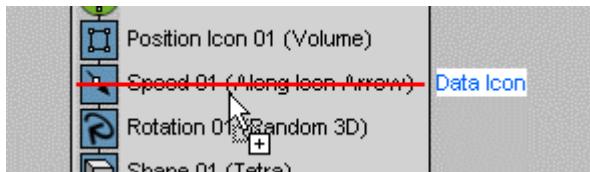
Autodesk® and 3ds Max® are registered trademarks of Autodesk, Inc.

Quick Start Tutorial

Part 1: Create a Data Operator

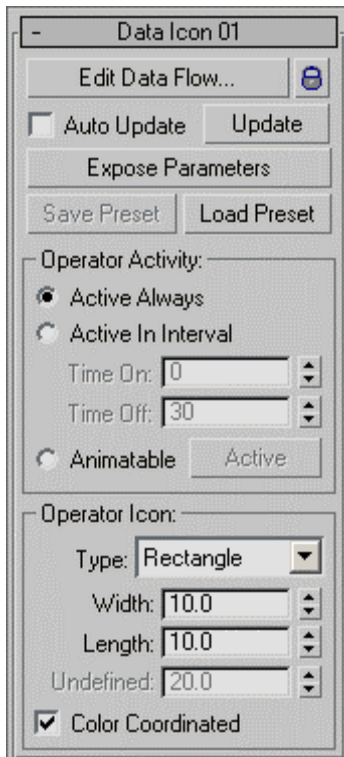
To create a new operator, you can use any of the Data operators: Data Icon, Data Operator, Data Test, or Data Icon Test.

1. Start or reset 3ds Max, and then create a default Particle Flow system. Press 6 to open Particle View.
2. From the Particle View depot, drag and drop the Data Icon operator on top of the Speed operator in Event01, so that when you release the mouse button, the Data Icon operator replaces the Speed operator.



3. Click the Data Icon 01 operator.

The Data Icon 01 rollout appears in Particle View.



4. At the top of the rollout, click the Edit Data Flow button.

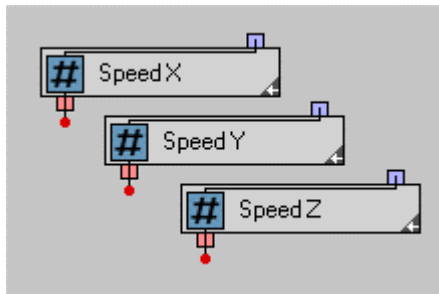
This opens [Data View](#), the user interface for Data operators.

For the first part of this exercise, you'll create a simple operator that defines particle speed with three numeric values. The creation and modification of particle properties is done by wiring different suboperators in Data View. In this example you'll use the Scalar suboperator.

5. From the Data View depot, drag and drop three Scalar suboperators – one for each vector component – into the main window.

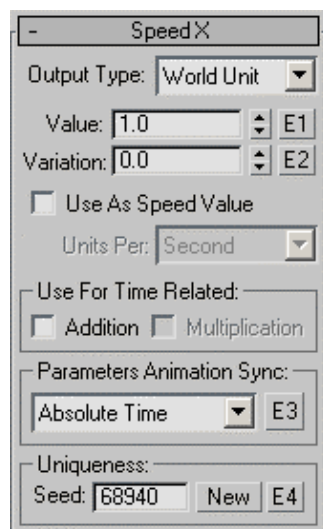
By default, the Dynamic Names feature is on, so each new suboperator assumes its most important setting or settings as a name. In this case, it's the Value setting: 1.0. You'll give the Scalar suboperators more meaningful names.

6. Right-click each suboperator in turn and name it **Speed X**, **Speed Y**, and **Speed Z**.



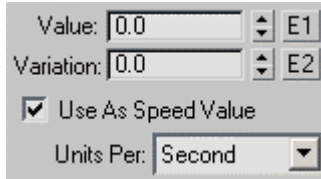
Next you'll define the initial values.

7. Click the Speed X suboperator.



Its parameters appear on the Speed X rollout.

8. Right-click the Value spinner to set it to 0.0, and then turn on Use As Speed Value. Keep the default Units Per setting: Second.



9. Do the same with the Speed Y suboperator.
10. Set Speed Z > Value to **200.0** and turn on Use As Speed Value.

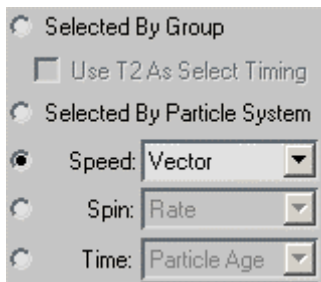
Next, you'll wire the data created by the Scalar suboperators into a speed data channel for the particles.

11. Drag an Output Standard suboperator into Data View, below the Scalar suboperators.

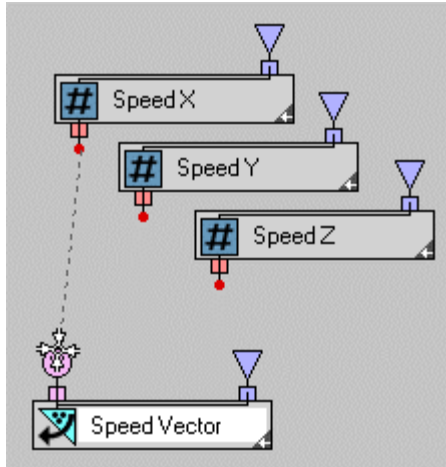


The purpose of the Output Standard suboperator is to take data calculated within the Data operator and convert it to information that the Particle Flow system can use. Thus, it has only an input; its output is meaningful only in the context of Particle View. Its default dynamic name is Position Vector because it's set to output position data in vector format.

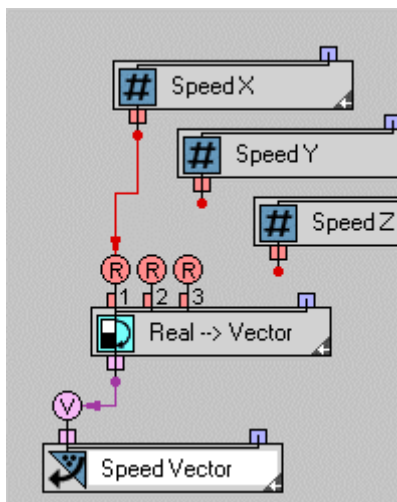
12. Click the Output Standard suboperator, and then, on its rollout, choose Speed, and set the drop-down to Vector.



13. Its dynamic name is now Speed Vector.
14. In the Data View window, drag from the output connector of the Speed X suboperator to the input connector of the Output Standard suboperator.

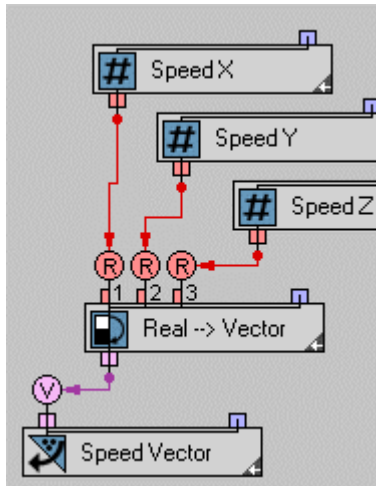


When you release the mouse button, a Convert suboperator automatically appears between the two suboperators you're linking. This is standard behavior in Data View when you attempt to wire together two connectors of different but compatible data types.



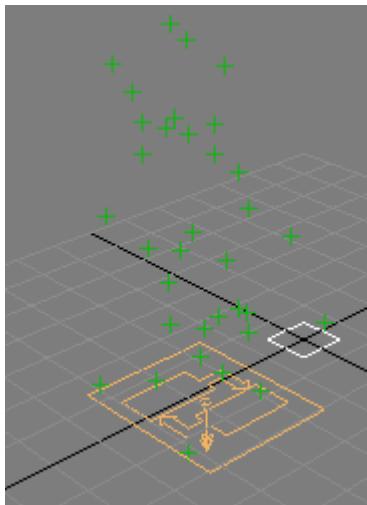
Thanks to dynamic naming, and the color coding of the inputs and output, you can see that the new suboperator is automatically set to convert real (scalar) data to a vector value.

15. Wire the output of the Speed Y suboperator to the second input (R2) of the Convert suboperator. Likewise, wire Speed Z to R3 of the Convert suboperator.



That's it! You've created a simple operator.

16. Scrub the time slider or play the animation. If necessary, move the dialogs to get a clear view of the viewport.



The particles move upward, in the positive Z direction, as instructed by your custom operator.

17. Go to frame 10, click the Speed Z suboperator in Data View, and then drag up and down on the Value spinner.

The particles move up and down to reflect the speed change.

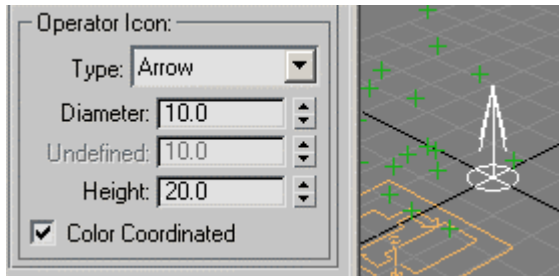
18. Click the Speed Y suboperator, and then change its value.

The particles move horizontally to reflect a change in speed along the Y axis.

Use the Icon to Control Particle Motion

Next, you'll make your operator more complex by linking the speed vector to the orientation of the operator icon. This is the small square at the world origin that was created when you added the Data Icon operator.

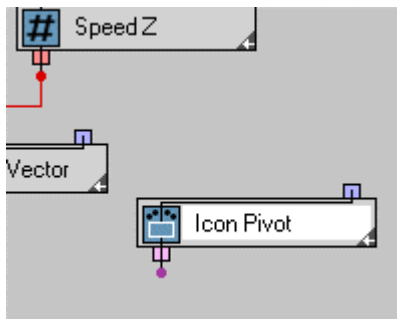
19. In Particle View, highlight the Data Icon operator, if necessary, and then in the Operator Icon group on the rollout set Type to Arrow.



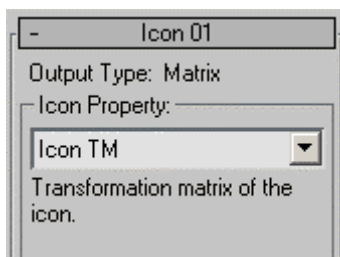
The icon changes to an arrow, which makes it easy to see its orientation.

You can use the Icon suboperator to request the transformation matrix of the icon.

20. In Data View, drag the Icon suboperator from the depot to an empty area of the schematic window.

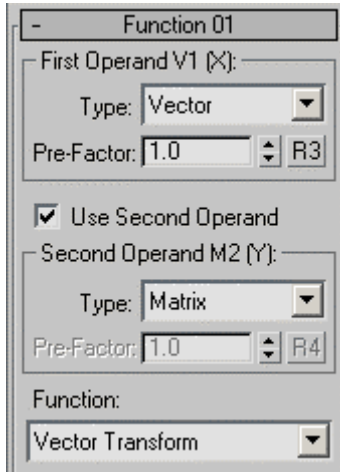


21. Click the suboperator and then, from the Icon Property drop-down list, choose Icon TM. TM stands for transformation matrix.



You'll use the Function suboperator to multiply the speed vector by the icon matrix.

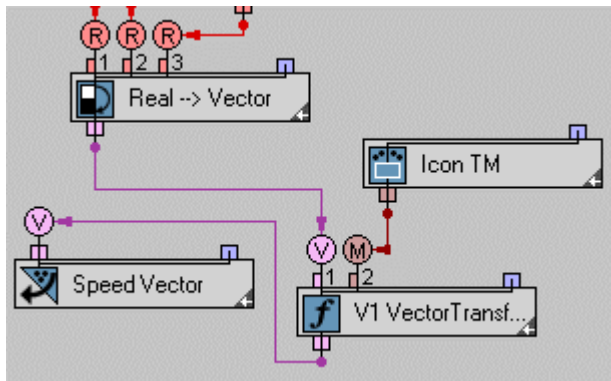
22. Add a Function suboperator in an empty part of the window.
23. Click the Function suboperator and then set the Second Operand > Type to Matrix. Also set Function to Vector Transform.



24. Click the wire between the Convert suboperator (Real --> Vector) and the Output Standard suboperator (Speed Vector) to highlight it and then press the Delete key to remove it.
25. Wire the output of the Convert suboperator to the V1 input of the Function suboperator.
26. Wire the output of the Icon suboperator to the M2 input of the Function suboperator.

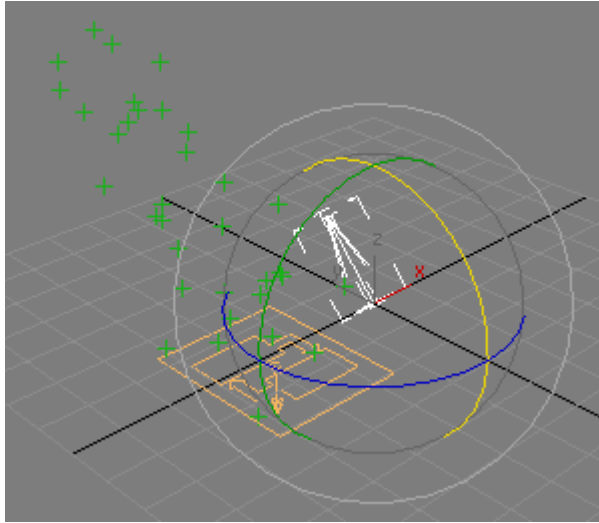
Note that you're wiring between the same-color connectors.

27. Wire the output of the Function suboperator to the input of the Output Standard suboperator.



Now the icon orientation controls the direction of the particles.

28. Rotate the icon about the X and Y axes.



As you rotate the icon, the direction of the particle stream changes accordingly.

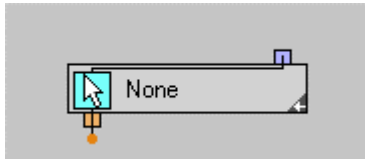
Replace the Icon with a Reference Object

It's just as easy to use a reference object's orientation to set the particle direction.

29. Add a teapot to the scene.

30. Click the Icon suboperator and then press the Delete key to get rid of it.

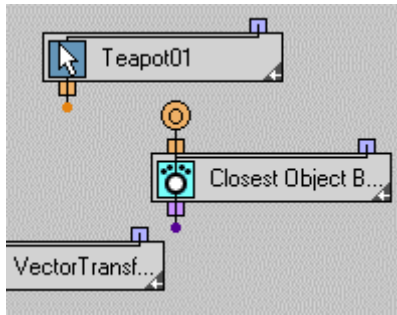
31. Add a Select Object suboperator to an empty area.



32. Click the Select Object (None), click the None button, and then select the teapot.

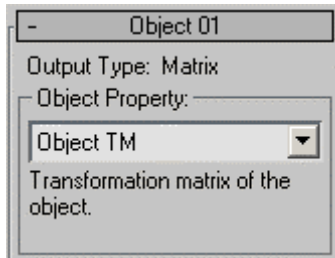
Its name appears on the button and on the suboperator.

33. Next, add an Object suboperator.

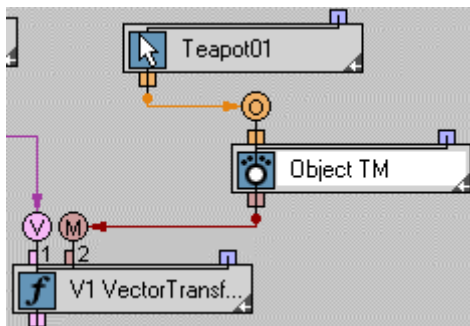


This lets you specify which object property to retrieve.

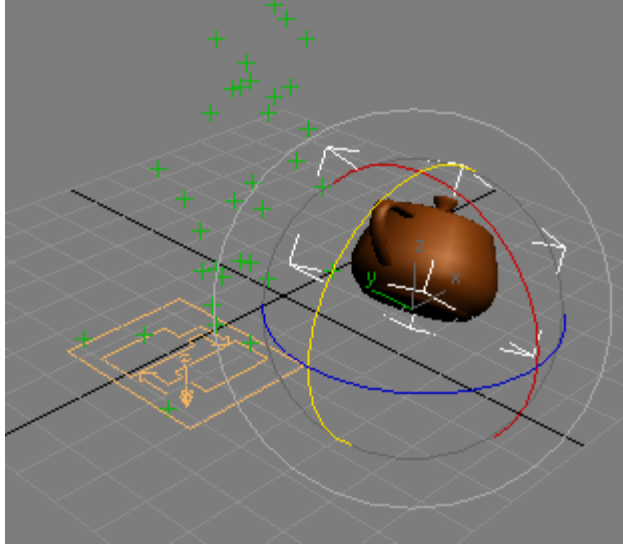
34. Highlight the Object suboperator and then, from the Object Property drop-down, choose Object TM.



35. Connect the Select Object suboperator to the Object suboperator, and then connect the Object suboperator to the M2 input of the Function suboperator.



Now, when you rotate the teapot about the X and Y axes, it has the same effect on the particle stream as did rotating the icon.



Part 2: Save the Data Operator as a Standard Operator

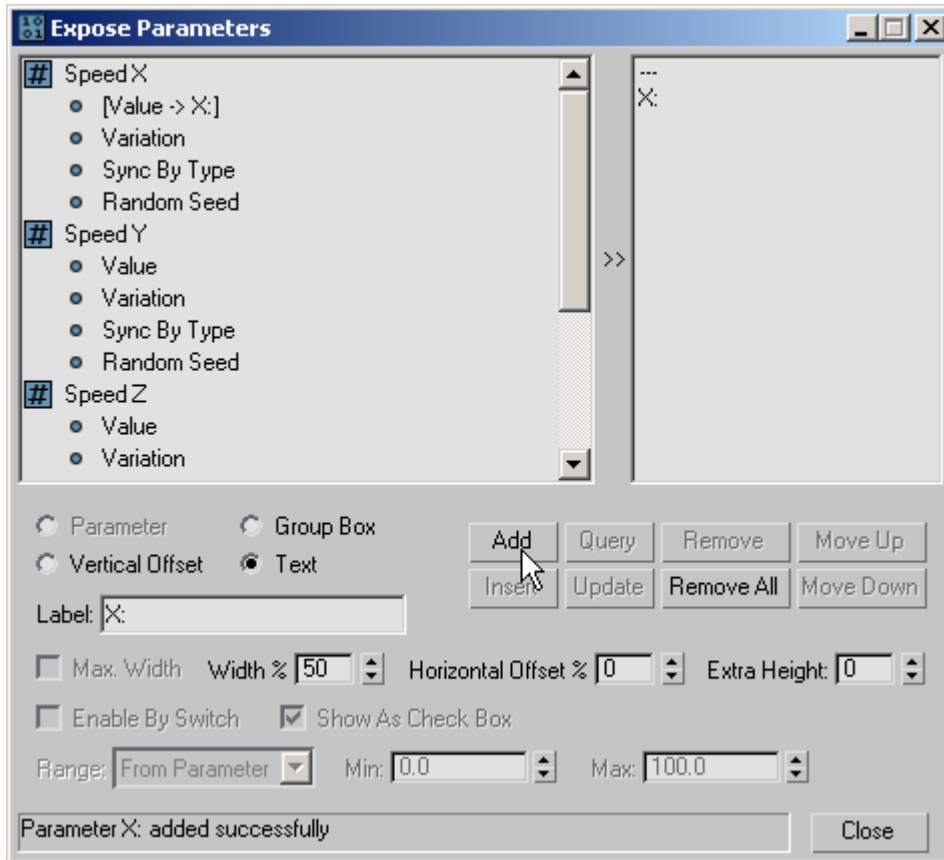
Once you've created a Data operator, you can save it as a regular operator. First, you expose the most important parameters. In this case, they're the vector components, that is, the Scalar suboperator Value settings, and the button of the Select Object suboperator.

1. In Particle View, make sure the Data Icon operator is highlighted, and on the rollout click the Expose Parameters button.

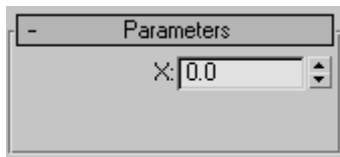
This opens the Expose Parameters dialog, which lets you collect and identify the parameters to be exposed. Basically, you use it to create your operator UI.

2. In the left-hand list, highlight the Value entry under Speed X. In the Label Field below the list, type **X:**, and then click the Add button.

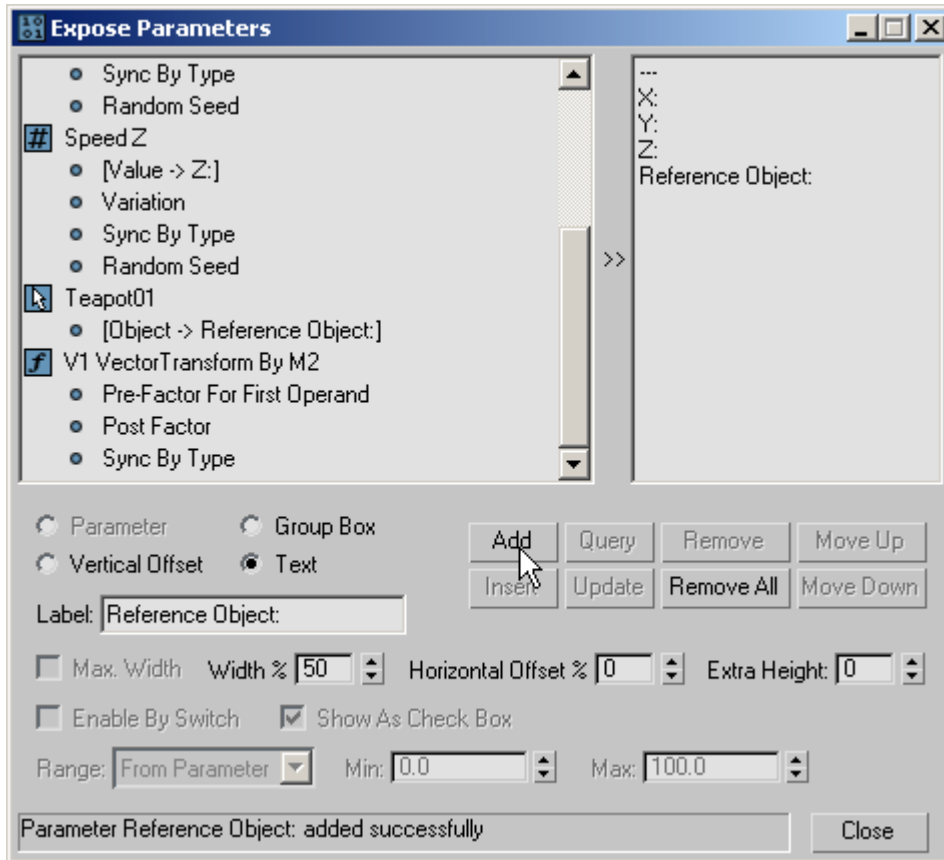
The entry in the left-hand list changes to [Value -> X:)], and the new label appears in the right-hand list.



More important, the added parameter appears on a new Parameters rollout in Particle View. This shows the exposed parameters, and is available to anyone who uses your operator.

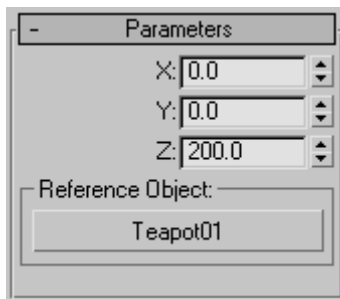


3. Do the same for Speed Y > Value and Speed Z > Value, labeling them Y: and Z:, respectively.
4. Also add Teapot01 > Object, labeling it **Reference Object:**. Also, before you click Add, turn on **Max. Width** so there's enough room for the button and label on the rollout.



5. On the Expose Parameters dialog, click the Close button.

All four parameters now appear on the Parameters rollout.



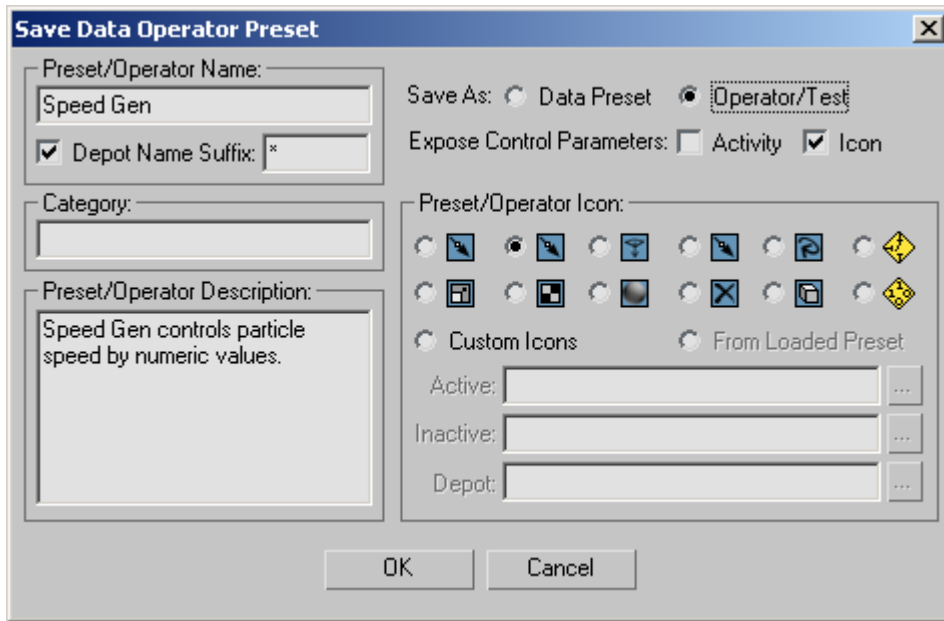
The next step is to save the Data operator as a preset.

6. On the Control rollout in Particle View, click the Save Preset button.

This opens the Save Data Operator Preset dialog.

7. In the Preset/Operator Name field, enter a name, such as **Speed Gen**.
8. To the right of this field, set Save As to Operator/Test.

9. In the Preset/Operator Description field, enter a description, such as this one:
Speed Gen controls particle speed by numeric values.
10. In the Preset/Operator Icon group, choose any icon you like.



11. Click the OK button.

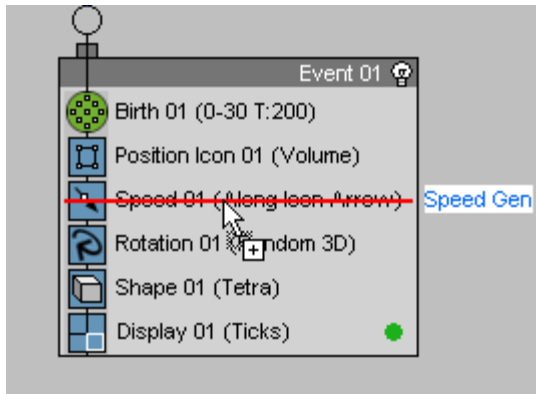
A message informs you that the preset has been saved.

12. In order to get the operator to show up in the Particle View depot, you need to quit 3ds Max and then restart. Do so now.
13. Add a PF Source object and a teapot. Press the 6 key to enter Particle View.

Now the Depot contains your new operator. The name of the operator in the Depot is followed by an asterisk (*), the default suffix, which denotes it as a custom operator.

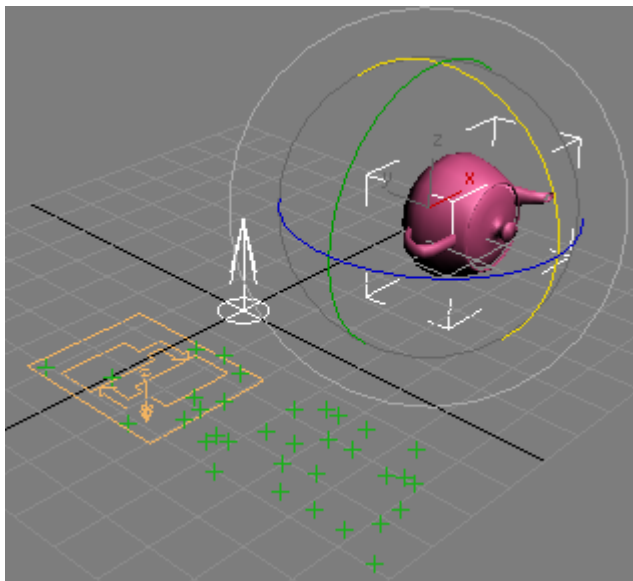


14. Drag the Speed Gen operator on top of the default Speed operator in Event01.

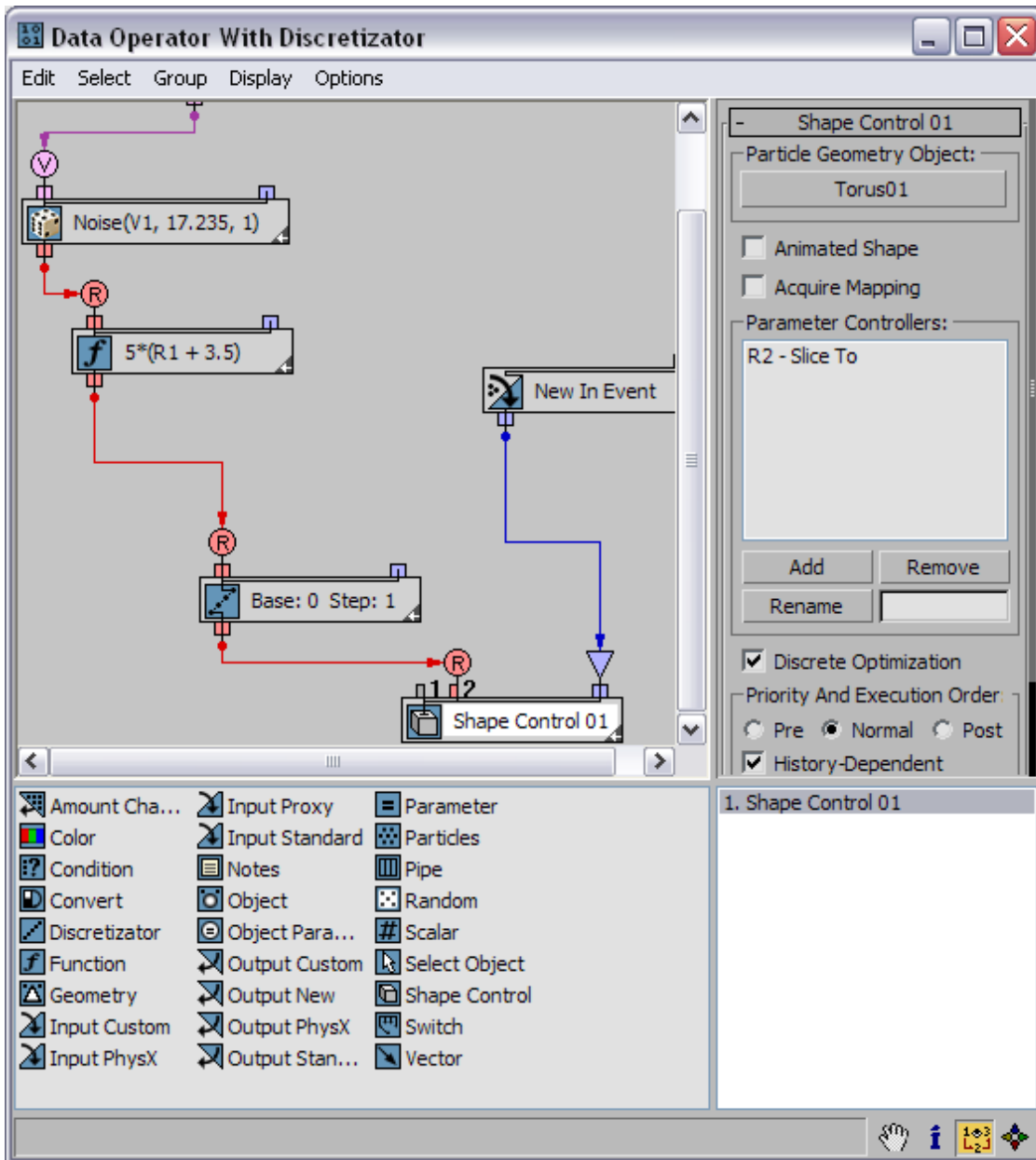


15. Click the Speed Gen01 operator, and then on the rollout in Particle View, click the Reference Object button and select the teapot.
16. Go to frame 10 or so and rotate the teapot.

As designed, the teapot orientation controls the particle stream direction.



Data View



Data View serves as the command center for *Particle Flow Tools: Box #3 Pro*. The basic operation of Data View is similar to that of Particle View: You start to build a data flow by dragging suboperators from the Depot at the bottom of the dialog into the main window. You access a suboperator's parameters by clicking it so its rollout appears on the right side of the dialog. And you create the actual flow, or schematic, by connecting suboperators, dragging between the output connector of one and an input connector or filter connector of another.

However, not all suboperators need to be physically connected; they can also be logically connected via Channel buttons, such as Particles suboperator > Weight Data Channel. In these cases you click the button and then choose the channel from the list in the Select Data Channel dialog that opens.

Wiring Suboperators

When connecting suboperators, it's important to keep the data types consistent. That is, the data type output by the "from" suboperator should be the same as the type input by the "to" suboperator. If the two data types are not the same but are potentially compatible, Data View automatically places a conversion suboperator between them when you connect them. For example, if you connect a suboperator that outputs vector data to one that inputs real data, Data View places a Convert suboperator between them, and sets it to Vector -> Real. Similarly, connecting a suboperator that outputs a data type other than Boolean to another suboperator's Filter input automatically inserts a Condition suboperator set to the correct input type between the two.

Note: When two operators are wired together and you change the data type of the wired input or output, the wiring is deleted. For example, if a Vector suboperator output is wired to an Output Standard suboperator set to Speed > Vector, and you change the Speed setting to Magnitude, Particle Flow removes the wire. Even if you then change the data type back to the previous setting, the wiring is not automatically restored; you must rewire the suboperators manually. Alternatively, using Undo restores the wiring.

Dynamic Suboperator Names

It's important to note that Particle Flow gives each suboperator a dynamic default name based on its most significant setting. For example, if you add a Particles suboperator, it's given the name Closest Particle Index because that's the default Aggregated Property setting. If you change the Aggregated Property setting, say to Density, the dynamic name changes to Density. You can turn off the Dynamic Name feature for each suboperator from its right-click menu (and globally using the Use Dynamic Names For New function on the Options menu). If you rename a suboperator, this automatically turns off Dynamic Name.

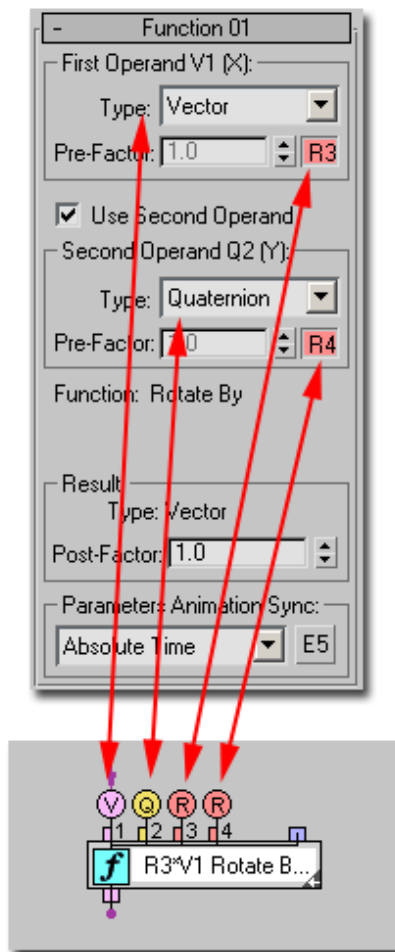
Suboperator Controls

The suboperator as visually represented in Data View is basically similar to an operator or test in Particle View. For example, you can Shift+drag a suboperator to create a copy or instance. You can click its icon to toggle its active status, drag its right edge to change its width, and right-click it to open a context menu of relevant commands.

However, there are several noteworthy functional differences. First, the suboperator has two types of inputs: data, of which there can be one or more, and filter, of which there is always one.

Data Inputs

The data inputs appear on the top of the suboperator, starting at the left side. Each input is numbered to correspond to the corresponding parameter in its rollout, and has two indications as to its data type: a letter and a color. For example, an input for vector data is violet, and for quaternion data is yellow. For a full list of the data types and their respective color codes, see [Data Types](#).



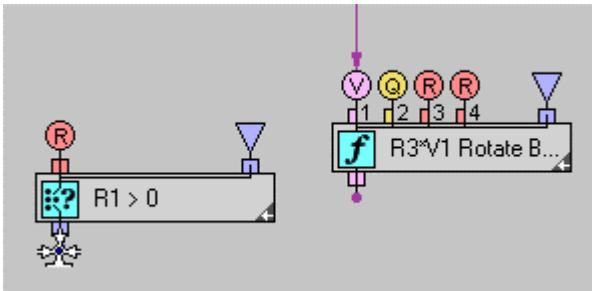
Suboperator with input connectors (below) and related parameters (above)

Filter Input

Near the top-right corner of each suboperator is the filter input connector; this lets you specify a subset of particles to process. The filter input accepts Boolean data only, as indicated by its blue color. Usage of this input is strictly optional; if you want the suboperator to process all particles in the event, you needn't use it. If you want the suboperator to process only certain particles, set up the condition and then wire it to the filter input. For example, you could limit processing to particles within a certain distance

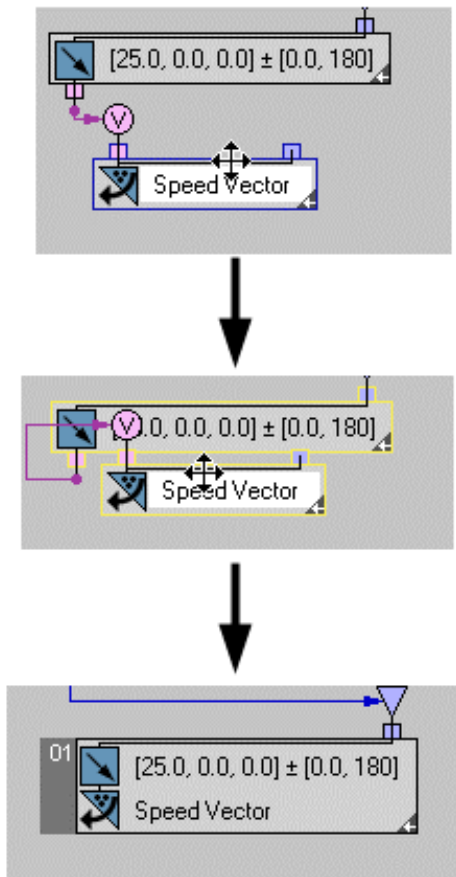
of the current particle, using a Particles suboperator wired to a Condition suboperator; you'd then wire the output of the Condition suboperator to the suboperator that is to perform the processing.

Note: When you position the mouse cursor over a suboperator's Boolean output connector, a funnel icon (an inverted triangle) appears temporarily over each filter input connector, as shown in the following illustration, to indicate that it can be wired to the Boolean output.



Suboperator Blocks

Data diagrams in Data View can quickly become very complex. To simplify the visual flow, you can combine suboperators physically into solid blocks. To do this, simply drag a suboperator on top of another to which it is wired. When yellow outlines appear around both suboperators, release the mouse to combine them into a block. You can remove a suboperator from a block simply by dragging it away from the block; release the mouse when the yellow outlining disappears.



Top: Start dragging a suboperator toward a connected one.

Center: When the yellow outlines appear, release the mouse button to combine them into a block.

Bottom: The block has a tab on the left side that you can use to manipulate it.

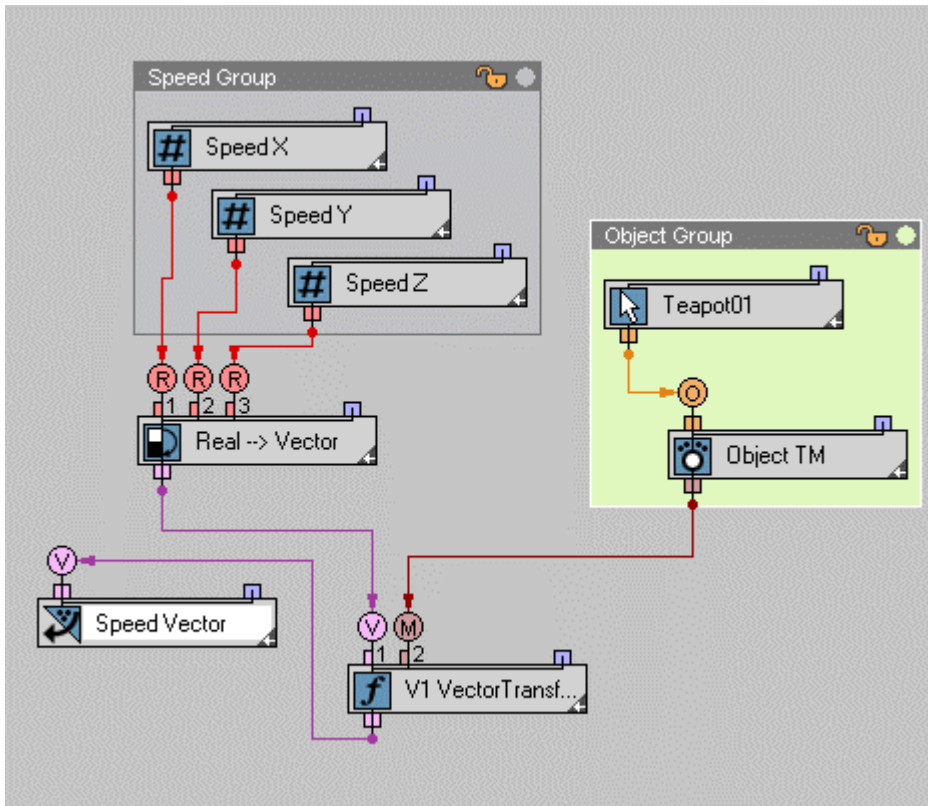
Important: You can combine two suboperators into a block only when the output of one is wired to the input of the other, and the latter has no other inputs. Effectively, this means that you can wire only single-input suboperators. As long as there is a continuous single-wire stream, you can combine any number of suboperators into a single group.

Grouping Suboperators

A different way of combining suboperators that has no logical restrictions is with groups. Grouping in Data View is similar to grouping objects in 3ds Max. To create a group, select some suboperators and then choose the Group command from the Data View Group menu or right-click menu.

Some additional points about groups:

- You can add a new suboperator directly to a group by dragging it into the group window from the Depot.
- If you drag a suboperator in a group toward the edge of the group window, the window expands to contain the suboperator.
- To place a group window in front of another group window, drag the former over the latter.



Groups in Data View

The controls, available on the Group menu, are:

Group – Combines all highlighted, ungrouped suboperators into a named group. Available only when more than one ungrouped suboperators are highlighted.

A group takes the visible form of a window containing the grouped suboperators, which you can reposition and resize like any standard window in Windows. The window title bar contains the group name and two icons: a lock icon and a small, circular color swatch. When locked, the suboperators cannot be moved within the group window, nor can they be resized. However, they can still be highlighted and their parameters edited. Use the color swatch to change the window background color.

Ungroup – Restores all suboperators in the highlighted group to their original, ungrouped condition. Available only when a group is highlighted.

Open/Close – Unlocks and locks the group; this is the same as clicking the lock icon in the group title bar. When a group is locked, you can move its window and change the member suboperators' parameters, but you can't resize the window or move the suboperators within the window.

Tighten – When on, the group window shrinks to fit a bounding rectangle around the member suboperators. When off, to resize a group window, drag an edge or a corner. If Tighten is on for a group and you enlarge the window manually, the software turns off Tighten for the group.

Transparent – When on, objects such as groups and suboperators are visible behind the frontmost group window.

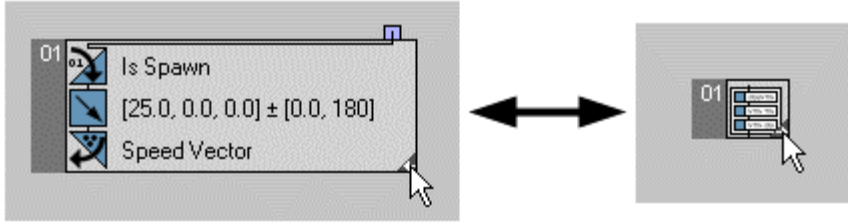
Attach – Adds the highlighted suboperator to the highlighted group. This command is available if the current selection set comprises a single open group and one or more suboperators, and for a suboperator if it overlaps an open group window.

Detach – Removes the highlighted suboperator from its group. The suboperator remains in place, but it's no longer part of the group.

The Group, Attach, and Detach commands are also available from the right-click menu in Data View for suboperators. Also, if you right-click a group in Data View, the menu provides the group commands Flip (flips the group window vertically, so the title bar is on the bottom), Tighten, Transparent, Open, and Close.

Additional Controls

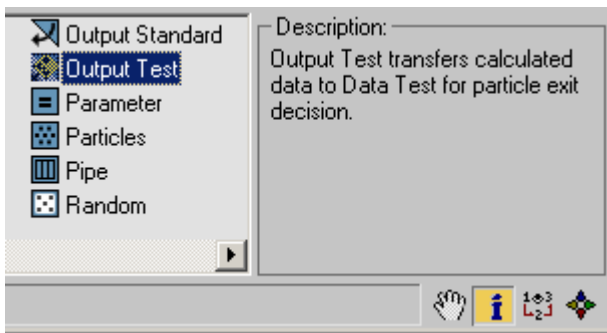
Each suboperator and block has a small arrow at its bottom-right corner. Clicking this arrow toggles the maximized/minimized state of the item. If it's maximized, the arrow points left; if it's minimized, the arrow points right. As with combining suboperators into blocks, minimizing helps reduce clutter in the Data View interface.



Left: Maximized group; Right: Minimized group

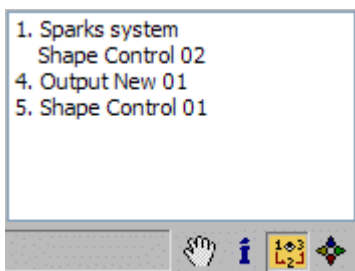
Also, in the bottom-right corner of the Data View dialog are found, in addition to the Pan tool present on the Particle View dialog, buttons for controlling the display.

The first button, Description, whose icon is a lower-case "i", simply toggles the Description box that displays information about the highlighted suboperator in the Depot.



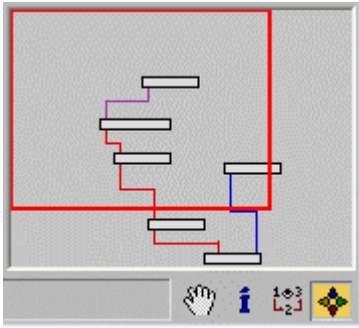
The second button, Output Order, shows the output order of suboperators that have Execution Order and Priority Order settings. If two or more suboperators have the same Execution Order value, they're listed under the same number, in the order of Priority Order values.

You can highlight the corresponding output suboperator in Data View for an item in the list by clicking the list member, and vice-versa (click the suboperator to highlight its list member).



The third button, Navigation, shows a mini-map of the Data View window. You can pan around the window by dragging inside the red outline, which shows the current extents of the window. Clicking outside the red outline jumps to that part of the window.

The Navigation map always shows the full data layout, so it zooms dynamically as you drag in the red outline. However, it does not affect the magnification of the main Data View window.



Menus

The Data View menus contain a number of standard, self-explanatory commands. This section covers only those that are unique to this software.

Edit menu

New – Choose a new suboperator to add from the submenu. The suboperator is added in an empty area of the Data View window.

Flip – Reverses the connectors of the highlighted suboperators vertically, so that inputs are on the bottom and outputs are on top.

Select menu

Select Outputs – Selects all output-type suboperators, such as Output Custom and Output New.

Group menu

For details on the Group menu functions, see [Grouping Suboperators](#).

Display menu

For details on the Display menu functions Description and Output Order, see [Additional Controls](#).

Options menu

Auto Update On Data View Close – The option is related to the [Auto Update option](#) in the Data Operator interface. When Auto Update is on, the changes in the suboperators of

the operator or test are automatically applied to the particle system. When off, you must click the Update button to apply the changes. When Auto Update is Off, you can open Data View and modify the suboperators as long as you like, without the delays caused by the particle system recalculations. When you are done with editing data flow in Data View, you can close it, and it will apply the changes automatically to the particle system, because the Auto Update On Data View Close option is on.

Use Dynamic Names For New – When on, the software uses dynamic naming for all new suboperators. This means that, when you add a new suboperator, the default name as shown on the suboperator in Data View is the most significant setting. So, for example, when you add an Object suboperator, by default it is named Closest Object By Pivot, because that's the default Object Property setting. If you then change the Object Property setting to Point Pivot, for example, the suboperator is renamed Point Pivot. This switch is on by default; if you turn it off, newly added suboperators are named with their type and a two-digit number (e.g., Object 02).

Convert All To Dynamic Names – Forces every suboperator to be named after its most significant setting.

Convert All To Custom Names – Forces every suboperator to be named with the name given it with Rename. If it was not renamed, its name is the suboperator type and a two-digit number.

Save Preferences – Saves the current layout and options as the default configuration. You're prompted to confirm this command.

Right-Click menu

Most of the functions in the Data View right-click menu are the same as in Particle View, and do not bear repeating here.

Data View Hotkeys

Several keyboard shortcuts, or hotkeys, are available for expediting your work in Data View. You can view and edit these in 3ds Max by going to Customize menu > Customize User Interface and choosing Group > Particle Flow Tools.

Note: The Move commands emulate the default WASD movement keys found in many first-person shooter games: W=up; A=left; S=down; D=right.

| Action | Default Shortcut |
|--------------------------------------|------------------|
| Copy Selected In Data View | Ctrl+C |
| Cut Selected In Data View | Ctrl+X |
| Paste In Data View | Ctrl+V |
| Rename Suboperator In Data View | F2 |
| Select All In Data View | Ctrl+A |
| Select None In Data View | Ctrl+D |
| Move Selected Down In Data View | S |
| Move Selected Down x10 In Data View | Shift+S |
| Move Selected Left In Data View | A |
| Move Selected Left x10 In Data View | Shift+A |
| Move Selected Right In Data View | D |
| Move Selected Right x10 In Data View | Shift+D |
| Move Selected Up In Data View | W |
| Move Selected Up x10 In Data View | Shift+W |

Data Types

Particle Flow Tools: Box#3 Pro uses a variety of data types; following is a comprehensive, color-coded list:

- Boolean – Blue
- Complex – Cyan
- Equal – Green
- Integer – Indigo
- Matrix – Maroon
- Object – Orange
- Pair – Pink
- Quaternion – Yellow
- Real – Red
- Time – Teal
- Vector – Violet

The colors are used by the input and output connectors in Data View, and, in certain cases, by UI buttons that enable additional inputs as alternatives to specifying values in the UI.

Most data types are self-explanatory. The following discussion covers the rest:

Pair and Complex Data Types

Some suboperators generate data that does not fit conveniently into the Vector, Integer, or Time format. Therefore, two new "artificial" types were added:

- **Pair** = { Vector + Integer }
- **Complex** = { Vector + Integer + Time } (not the traditional complex number).

Following is a discussion of situations in which the Pair and Complex types are used:

Object suboperator – Point Position option

One of the inputs is of the Pair type = { Vector + Integer }, where the Vector input is the position in local coordinates of the object and the Integer input is the index of the object, as defined by the Select Object suboperator. The object index is needed because the Object suboperator can work with multiple objects at once. If there is a single reference object to work with then the Integer value should be set to 0. Also, as a reminder, indices in Data View are 0-based (start counting from 0). This is different from MAXScript, where indices are 1-based. Therefore, if you have several reference objects as defined in the Select Object suboperator, then their indices are 0, 1, 2, etc.

Geometry suboperator – Closest Point option

The output is of the Pair type = { Vector + Integer } where Integer is a compound index that contains an object index and a face index, and Vector is the position in local face coordinates. When the closest point is calculated, the operator searches through all the reference objects, as defined by the Select Object suboperator, and finds the closest face as well as the closest point on the surface of this face. The local face coordinates use the edges of the face as the base vectors.

Geometry suboperator – Collision Point option

The output is of the Complex type = { Vector + Integer + Time } where Integer and Vector have the same meaning, as in the above instance, and Time is the time of the collision.

Geometry suboperator – Face Area and Face Selection options

The option can be used to calculate area of a face of an object (or the selection status of a face). And it looks like it is sufficient to supply the compound index as an index (object index + face index). However, the suboperator has a Pair-type input. This is done to simplify the wiring from the Closest Point option that has Pair as an output. If the object index + face index are created in a different way then you can use a Convert suboperator to create a Pair type; just use a zero vector as the other component of the Pair type.

Geometry suboperator – Point Color, Point Color Gradient, Point Self-Illumination, Point Mapping, Point Mapping Gradient, Point Material Index, Point Normal, Point Opacity, Point Position, Point Soft Selection, Point Speed options

One of the inputs is of the Pair type = { Vector + Integer } where Vector and Integer have the same meaning as in the Closest Point option.

Geometry suboperator – Random Surface Point option

The output is of the Pair type; the same meaning as in the Closest Point option.

Geometry suboperator – Random Volume Point option

The output is of the Pair type = { Vector + Integer } where Vector is the position in world coordinates and Integer is the index of the object used as a volume space.

Equal Data Type

The only suboperator that produces the **Equal** data type is the [Parameter suboperator](#). The Parameter suboperator is particularly useful for setting (or letting the user set) the same parameter value for several suboperators. Very often this common parameter is exposed later in the interface.

One simple example: You would like to make a Data Operator that places particles randomly on the surface of an object, and sets a random initial speed. You'll use the Geometry suboperator (Random Surface Point) for position, and the Vector suboperator to define the speed with Deviation set to 360 degrees for a random spread of direction.

Both suboperators have the Random Seed parameter. So, naturally, you would like to expose the Random Seed parameter so the user can play with chaos setup. But it might seem strange to have two Random Seed parameters in the interface. Therefore, you can wire both Seed parameters in the Geometry and Vector suboperator into a single Parameter suboperator with the type Uniqueness Seed. You can then expose Random Seed of the Parameter suboperator.

The Equal data type has subtypes: Real, Integer, Time, Sync, and Seed. Each uses a different shade of green. The Parameter suboperator has several output types: Angle, Float, Percent, World (all are Real type), Integer, Time, Animation Sync (as Sync) and Uniqueness Seed (as Seed). Angle, Float, Percent and World just differ as they are shown to the user in the UI, they are all Real. You can wire the Parameter output only to the matching E inputs, which is usually quite obvious--if you link the Random Seed of a suboperator then the Parameter suboperator should have Uniqueness Seed type as well.

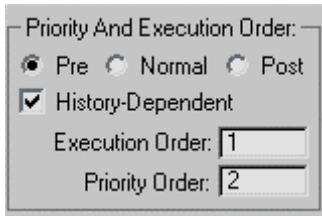
It is possible to live without the Parameter suboperator by using the standard 3ds Max Max script-wiring technique, and script-wiring the parameters of the suboperators, instead of linking them to a Parameter suboperator. However, script wiring is more time-consuming to set up, it does not have good visual reference as the Parameter suboperator, and it is prone to save/load and clone problems.

Compound Index

Sometimes there is a need to pack two integer values into a single value, when, for example, an integer is used as a compound index = { object index and face/vertex index }. You can use the [Convert suboperator](#) to a) create a compound index out of to integer data; or b) split object index or face/vertex index out of a compound index. A compound index is not a new data type--it's a way to pack more information into an integer data channel.

Priority and Execution Order

The Output-type suboperators, including Amount Change, have options to define the Priority order and Execution order. When creating complex setups, it's important to pay attention to these options.



The order of the data modification and processing is defined by the Output suboperators. The Output suboperators drive data modification. They pull particle data out of suboperators above them in the data flow, and set the data as particle properties.

Output suboperators with the same Execution Order value deal with the same input data. When the Execution Order value is the same, the Priority Order value defines the order in which the Output suboperators pull the data from upstream. Only after all the data are pulled are the data applied to particle properties.

If suboperators have different Execution Order value then they might deal with different input data, even if they pull the data from the same Input suboperator. This is because an Output suboperator with a lower execution order might change the data in some particle property. When the output suboperator with higher execution order pulls the data, these data have been already modified.

Consider this example: When particles come into an event, you need to calculate initial values for all new particles. Then you want to modify particle properties in every frame (regardless if a particle is new in event or not), and during this modification you need initial values to be set in all particles.

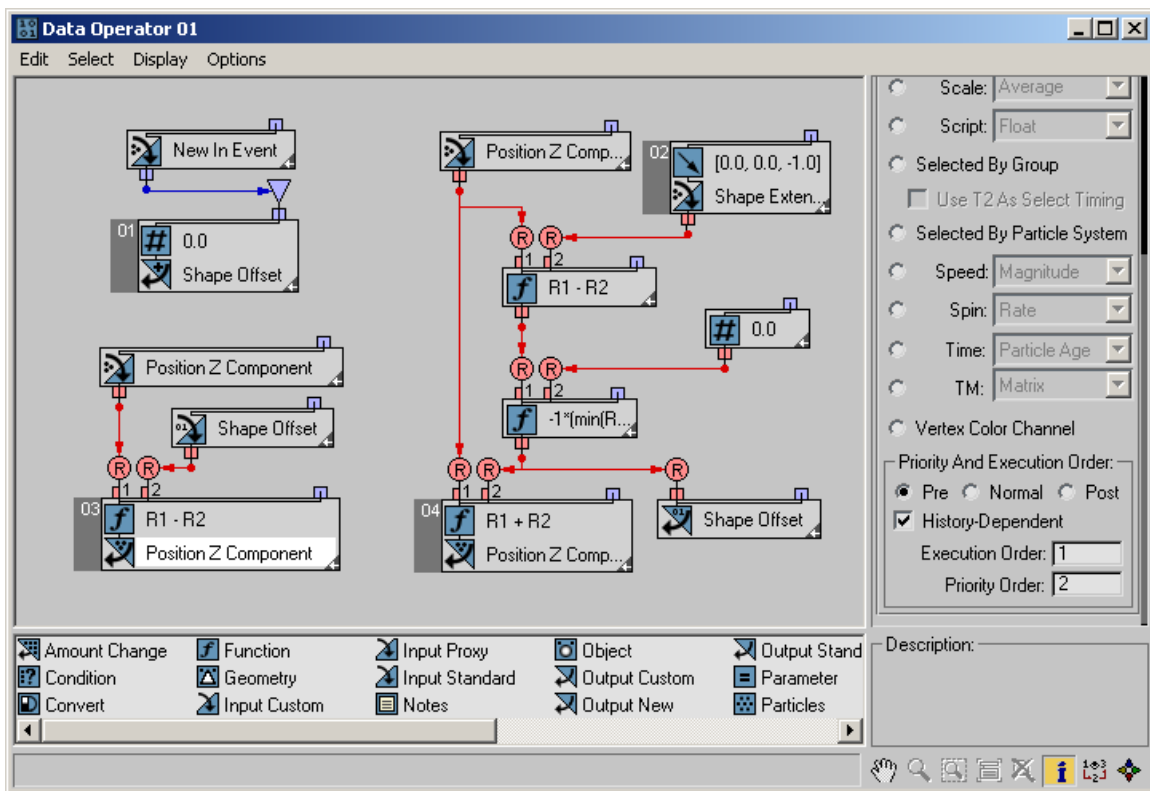
For this scenario you have two data streams: the first stream calculates the initial values; and the second stream modifies particle properties. The Output suboperator for the first stream should have Execution Order=1, and most likely its filter input will be set to Input Standard > New In Event. The Output suboperator in the second stream should have Execution Order=2.

Consider another scenario: You want to analyze speed data in all particles, and, depending on some condition, you would like to define two subsets of particles, each of whose speeds will be modified in different ways. For this scenario you have to use two Output suboperators with the same execution order. This way you guarantee that the initial input speed values are the same for each Output.

Incidentally, because the Amount Change suboperator changes the number of particles, it cannot share its Execution Order value with any other suboperator. Safeguards in the program prevent you from doing so.

Also, the Execution And Priority Order group also has Pre/Normal/Post radio buttons. This option defines the processing order on a higher level. If it set to Pre then this output is done before any operators in the current event are executed. If it is set to Normal, then the output suboperator is executed in the regular order, in-between the other operators in the same event. If it is set to Post then the output is processed after all operators and all tests in this particle system have finished their jobs for this integration step.

For an example that illustrates Pre/Normal/Post usage, open the included file *CollisionAsBody.max*.



This example uses the Data operator to produce "cheap and quick" full-body collision. As you probably know, particle systems in 3ds Max treat particles as point entities with regard to collisions and deflectors. Therefore, if a particle has a shape with visible size, the shape penetrates the deflector surfaces, which can ruin the overall visual effect. The example shows how to wire particle data to offset the particle position by the amount of the inter-penetration.

First, we create a custom data channel to store real values, as the position offset needed to avoid the penetration. For all new particles we initiate this value to zero--see data block 01.

After all the other operators and tests have done their job, we calculate the amount of penetration, and offset particle position in Post phase. This way particle positions are adjusted to avoid penetration and the particles are ready for rendering. The Output operators on the right side do their jobs in the Post phase.

Then, in Pre phase, we need to move particles back, to their no-offset positions, for the collision tests to continue their job as if nothing has happened.

History-Dependent

Particle systems are inherently history dependent. In other words, to be able to determine the state of a particle system at, say, frame 100, the software has to calculate the states for all previous frames since the first particle is born. This is because the changes in particle properties--for example, the positions of the particles--accumulate due to the constant changes in other parameters, such as speed.

However, for particular setups some particle properties are not history dependent, and can be calculated from the current state of the particles. For example, the orientation of the particles if they "look at" something such as the camera, or follow particle speed. Here the orientation of the particle depends either on particle position, as is the case when facing the camera, or particle speed. In this case, to calculate the current orientation of the particles, you don't need to know the entire history of the particles; you just need to know their current speed or position.

In that sense, the operator that calculates particle orientation is not history dependent. The operator does not have to do anything while the system runs through all prior frames in order to get to the current frame. The operator has to work only on the final frame; to analyze the position/speed of the particles in the final frame and calculate the corresponding particle orientation.

If you are sure that the result of the Output suboperator does not depend on the history of particles, you can turn off the History-Dependent option. This can speed up the overall calculations, as it causes the Data operator to work at the last frame only.

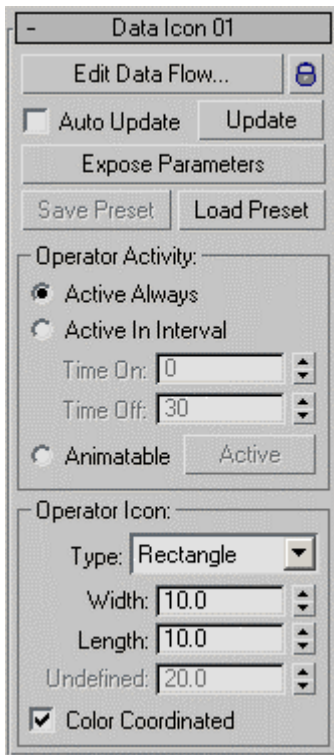
Data Icon/Operator/Icon Test/Test

Path: Particle View > Add or click Data Icon/Operator/Icon Test/Test.

This topic documents the parameters for the Data Operator, Data Icon operator, Data Test, and Data Icon Test actions. The basic parameters are the same for all four actions; the only difference is that the Icon versions have additional parameters for controlling the icon.

Note: For Data Icon and Data Icon Test, the parameters rollout is also available on the 3ds Max Modify panel by selecting the icon.

Interface



Edit Data Flow – Opens [Data View](#), where you can build and edit a data flow for the current operator or test. If the operator/test is locked (see following), you must click the lock button to the right and enter the correct password before you can click Edit Data Flow.

[lock button] – Lets you add password access to the operator/test.

When this button is off, click it to open the Enter Password dialog, and then enter a password and click OK or press Enter. The dialog displays the current password, or, if you haven't set a password previously, a default password.

When on, clicking the button opens the Enter Password dialog; enter the password to gain access to the settings on the upper part of the rollout. If the wrong password is entered, these settings remain unavailable. Note that password access is case sensitive.

Auto Update – When on, changes to the operator or test are automatically applied to the particle system. When off, you must click the Update button to apply changes.

Update – Applies changes to the operator or test.

Expose Parameters – Makes internal parameters available to direct control by the user. Opens the [Expose Parameters dialog](#), described below.

Save Preset – Lets you save the current Data operator/test as a preset, which you can then use as a regular operator from the Particle View depot, or add with the Data Preset operator or test, or with the Load Preset function. Opens the [Save Data Operator Preset dialog](#), described below.

Save Preset is available only when a valid operator is present.

Load Preset – Applies an existing preset to the current Data operator/test. Opens the Select Data Preset To Load dialog. In the dialog, click the preset to load, and then click OK. The preset's parameters are applied to the Data operator/test.

The Select Data Preset To Load dialog respects the current Data action type. When opening a preset from a Data operator, it shows only presets saved from operators, and likewise for tests.

Operator Activity group

This setting determines when the operator or test is active. The choices are:

- **Active Always** – The operator/test is active throughout the animation. This is the default choice.
- **Active In Interval** – Lets you set a single interval, in frames, during which the operator/test is active. Use the Time On and Time Off parameters to set the interval.
- **Animatable** – Lets you set multiple intervals during which the operator/test is active, via keyframing and the Active button. Turn on Auto Key, go to the frame at which the active status should toggle, and then click the Active button. Repeat as necessary.

Operator Icon group

This setting, available only with the Data Icon operator and Data Icon Test actions, sets parameters for the icon associated with the action.

Type – Lets you set the icon shape. Choose a shape from the drop-down list. The shape can be useful when using [Icon suboperator](#). The Icon suboperator can query different properties of the operator icon.

The remaining parameters in this group are determined by the Type choice. For example, Rectangle, the default type, lets you set Width and Length. These parameters are self-explanatory.

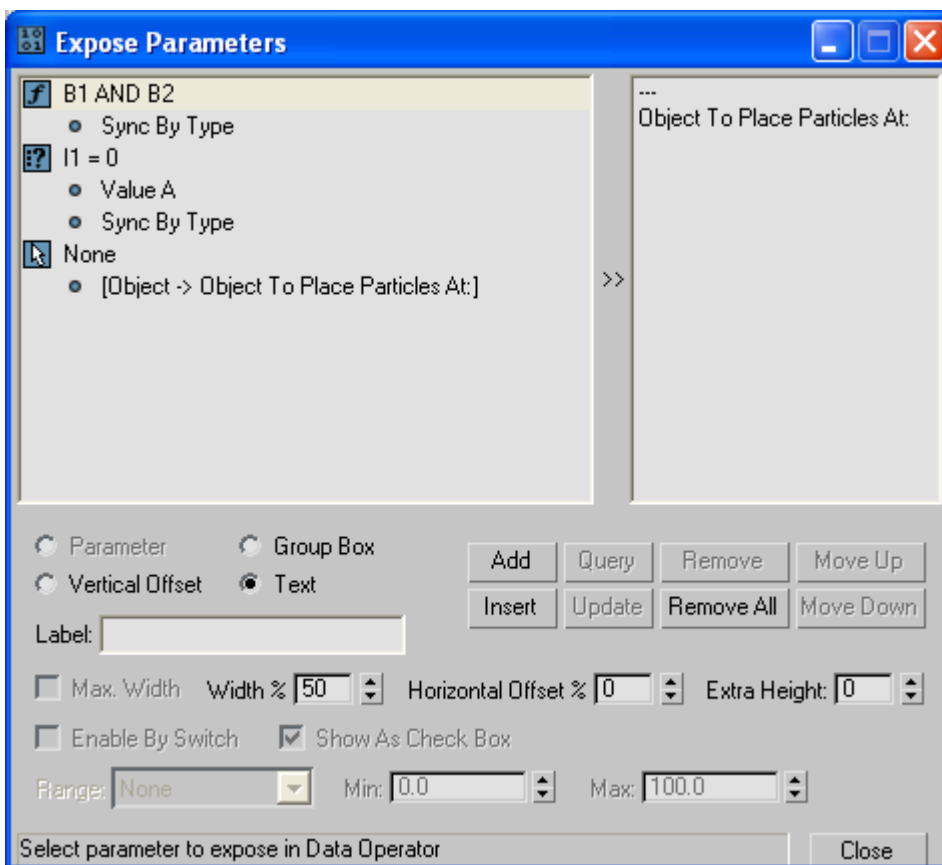
Expose Parameters Dialog

Path: Particle View > Add or click Data Icon/Operator/Icon Test/Test > Click Expose Parameters.

The standard procedure when exposing parameters is to highlight a parameter in the left-hand column, set the parameters in the lower part of the dialog, and then click Add.

For an example of how to use Expose Parameters, see [Save the Data Operator as a Standard Operator](#).

Interface



[control type] – The radio buttons above the Label field let you define explicitly the type of control to be exposed. The available radio buttons in this set depend on the type of control highlighted in the list. The choices are:

- Parameter
- Group Box
- Vertical Offset
- Text

Label – The name the exposed parameter will have in the operator UI in Particle View.

Max. Width – Applies to drop-down list and button control types. When on, creates a frame box for the control instead of a text label on the left side.

Width % – Sets the horizontal size of the parameter with respect to the width of the Parameters rollout.

Horizontal Offset % – The distance of the parameter from the right side of the rollout. In some cases you'll need to adjust the Width % and Horizontal Offset % values after exposing the parameter.

Extra Height – Adds vertical distance, in pixels, between the parameter and the one below it.

Enable By Switch – The [Switch suboperator](#) can pass along any of several different data streams, depending on its Active Input value. It might happen that some exposed values are taken from alternate data, upstream. In such a case, if, due to the Active Switch value, the data are taken from a different stream, the exposed parameter is not used, so, for clarity, you can gray them out. To tell the UI about this situation, when exposing such a parameter, you can turn on Enable By Switch. Then, after clicking the Add button, the user is asked to choose a Switch suboperator.

Show As Check Box – Lets you choose how to expose the active input of a Switch suboperator. Available only if the Switch suboperator has two inputs. In this case, when on (the default), the control appears as a check box. When off, the control appears as a drop-down list.

Also see the Max. Width option, above.

Range – Choose whether the permissible parameter range is:

- **None:** Unlimited range
- **From Parameter:** As specified by the settings in Data View
- **Custom:** Lets you set the range explicitly, using the Min and Max settings.

Min/Max – When Range is set to Custom, use these to set the minimum and maximum values in the range.

Add – Exposes the parameter. Click Add to place the parameter highlighted in the left-hand column at the end of the list in the right-hand column and on the Parameters panel, using the specified settings.

Note: After you add the first parameter, a "---" entry automatically appears at the beginning of the list. This represents a blank line above the topmost parameter.

Query – Updates the dialog settings with the values from the highlighted parameter in the right-hand column.

Remove – Deletes the highlighted parameter in the right-hand column.

Move Up – Moves the highlighted parameter in the right-hand column one position higher.

Insert – Adds the parameter above the highlighted position in the list, rather than at the end.

Update – Revises the highlighted parameter in the right-hand list, using the current settings. To use this, highlight a parameter, click Query if necessary to get the current settings, change the settings, and then click Update.

Remove All – Clears the right-hand column and deletes the Parameters rollout.

Move Down – Moves the highlighted parameter in the right-hand column one position lower.

Save Data Operator Preset Dialog

Use this dialog to save your Data operator or test as a preset or operator; in the latter case, it appears in the Particle View depot *after you quit and restart 3ds Max*.

For an example of how to use Expose Parameters, see [Save the Data Operator as a Standard Operator](#).

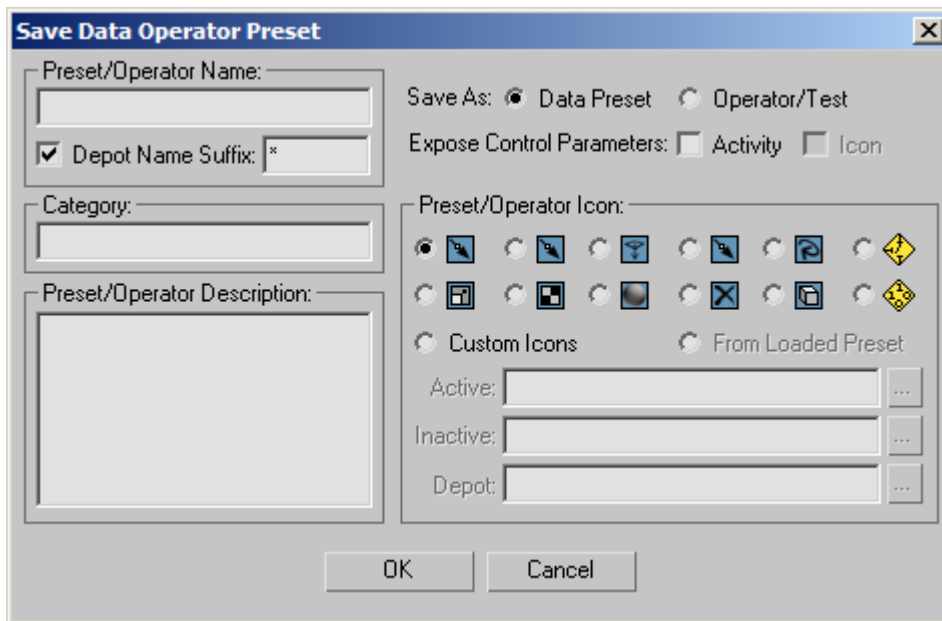
Procedure

To delete and organize saved operators and presets:

A saved preset or custom operator takes the form of a file with the extension *.pfp* (Particle Flow Preset), saved in the folder *[3dsmax root]\plugcfg\Particle Flow Data Presets*. To remove a preset or operator, navigate to this folder and delete the file.

If you save a preset in a particular category, the software creates a subfolder with this name, and places the preset there. By reorganizing folders in the *Particle Flow Data Presets* folder, you can reorganize the category organization for the presets.

Interface



Preset/Operator Name – The name the preset or operator will have in the Select Data Preset To Load dialog or the Depot, respectively.

Depot Name Suffix – Enter one or more characters to be appended to the Depot name to distinguish the custom operator or test from the rest. The default is *. This applies to presets saved as Operator/Test only.

Category – Applies only to saving presets. When you specify a category name, the preset appears in a branch of that name in the Select Data Preset To Load dialog.

Preset/Operator Description – The description you enter here appears on the Description panel to the right of the Depot.

Save As – Specify whether to save your custom operator/test as:

- **Data Preset** – The Data operator/test is available as a preset within Data View, and from Particle View with the Data Preset operator or test.
- **Operator/Test** – The Data operator/test is available in the Particle View depot after you quit and restart 3ds Max.

Tip: To create a blank line in the description, press Ctrl+Enter, or simply enter an appropriate number of spaces. Pressing Enter by itself accepts any changes and closes the dialog.

Expose Control Parameters – A Data operator, besides the wired network of suboperators, has some common parameters that are shown in its rollout in Particle view, in the Operator Activity and, for Data Icon and Data Icon Test, Operator Icon groups. If these parameters should be shown in your custom operator, you can turn on Activity or Icon or both. If you don't expose these parameters, then only controls you expose explicitly with [Expose Parameters](#) will be available.

Preset/Operator Icon group

Use these controls to specify an icon or icons for your custom operator or test. Choose one of the predefined Active icons (each also has an accompanying Inactive and Depot version), or specify images for a custom icon.

Custom Icons – Choosing this activates the Active/Inactive/Depot fields. Use the [...] buttons to choose image files in BMP format for when the operator/test is active, inactive (turned) off, and as it appears in the Depot (when saving as operator/test).

From Loaded Preset – Uses the icon from the currently loaded preset, if any.

Amount Change Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Amount Change.

The Amount Change suboperator lets you modify particle count by spawning new particles or deleting existing ones. When spawning, you can record spawning properties and details into custom channels created with the Output New suboperator. Also when spawning, the value of the Integer input determines the number of new particles.

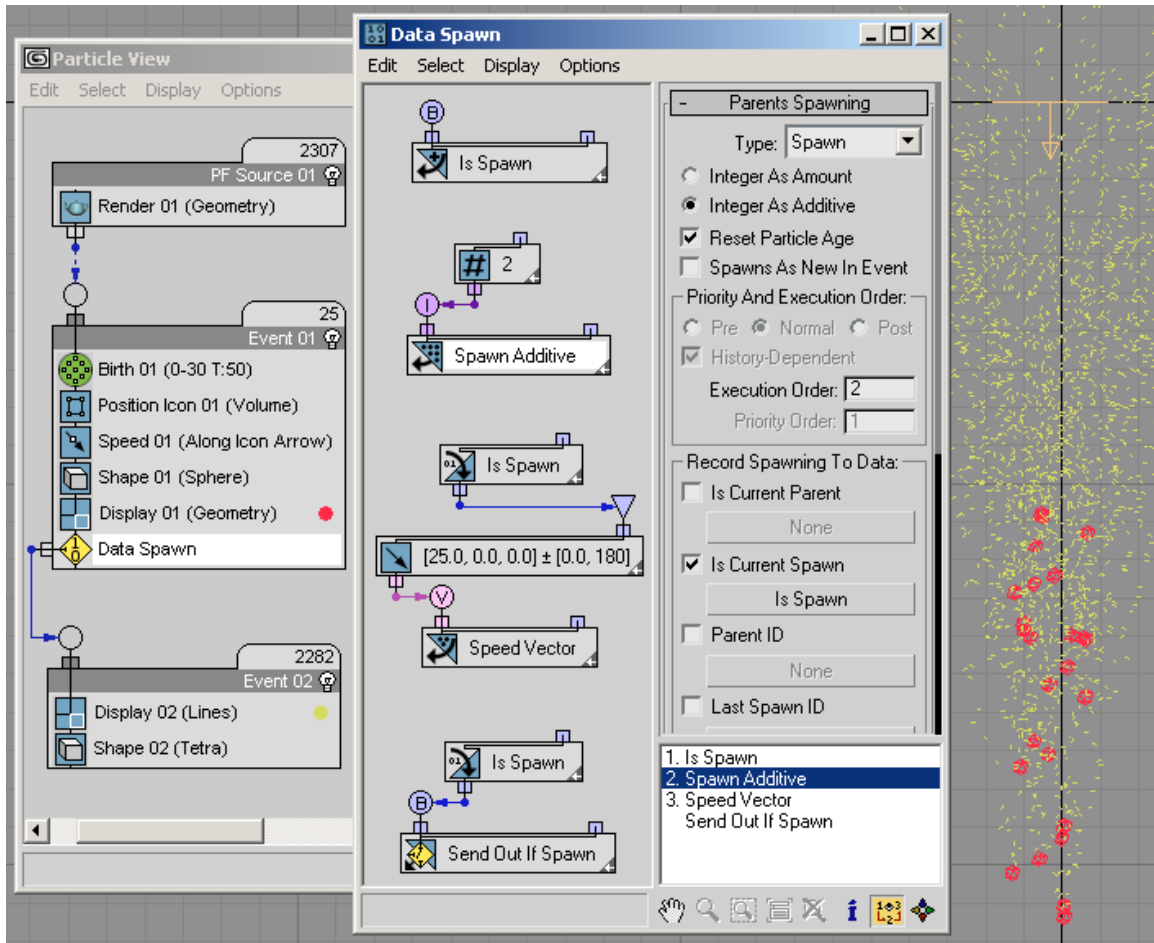
Procedure

Amount Change is one of the most complex suboperators in *Particle Flow Tools: Box#3 Pro*. To help you understand how it works, we offer the following example: Simple Spawn.

Use of the Amount Change suboperator requires knowledge of the other suboperators. The purpose of this example is just to illustrate the simplest way of using the Amount Change suboperator.

Note: In this scene, the original (parent) particles move along the PF Source icon arrow, as defined by the standard Speed operator, while spawned particles move in various directions.

To follow along, open the included file *SimpleSpawn.max*.



1. Add an Amount Change suboperator. Leave it set to Type: Spawn and choose Integer As Additive. Next, wire its input to a Scalar suboperator set to Output Type: Integer. The integer value of the Scalar suboperator controls the intensity of the spawning process.
2. Create a custom channel by adding an Output New suboperator with Data Type: Boolean to record the spawning.
3. Set the Output New > Execution Order to 1. This automatically sets the Amount Change > Execution Order to 2, so the custom channel is created before Amount Change is executed.
4. In the Amount Change suboperator > Record Spawning To Data group, turn on Is Current Spawn. Click the None button and choose the Output New suboperator. This records the spawned data to the custom channel.
5. To see the spawned process, change the speed of the spawned particles (speed vector= [25, 0,0], Divergence=180) and turn on Use As Speed. The Vector suboperator used here with Divergence separates individual spawn particles from each other and the parent.
6. (necessary for Data Test) To filter spawned particles from the original ones, add an Output Test suboperator. The Output Test should request the data from the custom channel; that is why a new Input Custom suboperator is created.

In its current form this example is just a subset of the regular Spawn test, from which you can build a custom spawn process. For example, instead of using the Scalar suboperator to define the additive amount for spawning, you could create your own schematics that calculate the spawning number based on the position of the parent. Another avenue for experimentation is making changes to the Vector suboperator (defining the speed of spawned particles).

This example shows the minimum requirements for a spawning process. The Amount Change suboperator alone is not enough for that. So, the purpose here is to give some basis and background to work with. Once you've laid the groundwork and seen that it's working (the spawning is visible), it is easier to modify or build on.

Interface

The image displays two screenshots of a software configuration interface for an event named "Amount Change 01".

The left screenshot shows the configuration for the "Spawn" type. The "Type" dropdown is set to "Spawn". Under "Priority And Execution Order", "Normal" is selected, and "History-Dependent" is checked. The "Execution Order" and "Priority Order" are both set to 1. The "Record Spawning To Data" section contains several unchecked checkboxes, each with a "None" button below it: "Is Current Parent", "Is Current Spawn", "Parent ID", "Last Spawn ID", "First Spawn ID", "Current First Spawn ID", "Older/Previous Sibling ID", "Current Spawn Count", "Total Spawn Count", and "Current Spawn Order".

The right screenshot shows the configuration for the "Delete" type. The "Type" dropdown is set to "Delete". Under "Priority And Execution Order", "Normal" is selected, and "History-Dependent" is checked. The "Execution Order" and "Priority Order" are both set to 1. The "Record Spawning To Data" section is not visible in this view.

Type – The type determines whether Amount Change creates (spawns) particles or deletes them. When Type is set to Delete, the only controls available are False To Delete/True To Delete and Execution Order. Default=Spawn.

Integer As Amount/Additive – The Amount Change suboperator's Integer data input defines the number of particles spawned. Use this option to determine whether the incoming data is regarded as:

- **Amount** – The input Integer value defines how many particles will replace the parent. That is, it defines the total number of particles, including the parent and its spawned offspring. An input value of 0 means that a parent particle is deleted; 1 means that the particle doesn't spawn, and so on.
- **Additive** – The input Integer value defines the number of spawned particles. An input value of 0 means that the particle doesn't spawn, a value of 1 means that 1 child is created per parent particle, and so on.

False/True To Delete – Available only when Type=Delete. This is the sole function of the Delete option, which sets the suboperator input type to Boolean: When incoming data is of the same value as the chosen option (False=0; True=1), particles in the event are deleted.

Reset Particle Age – When on, particles in the event have their age reset to 0.

Spawns As New In Event – All particles have a standard "New In Event" data channel that indicates particles that have just entered the event, either by birth or by transferring from a different event. The "New In Event" data is used by many operators to initialize some property data for particles. This option lets you choose whether to tag newly spawned particles with this data channel. When on, a spawned particle is considered new, thus triggering some data initialization by other operators. When off, a spawned particle is not considered as new, leaving these data to be inherited from the parent particles.

Priority And Execution Order group

See [Priority And Execution Order](#).

Record Spawning To Data group

These controls let you record spawning properties and details into custom channels. The custom channels have to be created in advance with the Output New suboperator. The data in the custom channels are updated during the Amount Change execution. Therefore, in order to obtain these data, you have to request the data from the custom channels (usually with the Input Custom suboperator) in the data stream with Execution Order value set higher than the Execution Order value of the Amount Change suboperator.

To use any of these controls, turn it on, click the associated button, and then use the Select Data Channel dialog to specify the channel.

Is Current Parent – Boolean data to indicate if a particle spawned any particles.

Is Current Spawn – Boolean data to indicate if a particle is a recent spawn.

Parent ID – Integer data with the birth ID of the parent of a particle. The ID information can be used in the Input Custom suboperator that can accept an integer channel for retrieving data from other particles (option Use I2 As Particle IDs). The datum reads -1 if a particle is not a result of spawning by the Amount Change suboperator. Incidentally, you can use the same data channels in between several Amount Change suboperators.

Last Spawn ID – Integer data with the birth ID of the last spawn particle. The data are valid only for particles that were parents at least once, and not necessarily in the current integration step but rather globally in time.

First Spawn ID – Integer data with the birth ID of the first spawn particle in the global sense. This particle is the oldest one among the spawns of the parent particle. Again, the data are valid only for particles that were parents at least once. The data channel is available only if the Total Spawn Count channel (see below) is defined.

Current First Spawn ID – Integer data with the born ID of the first spawn particle in the current integration step.

Older/Previous Sibling ID – Integer data with the birth ID of the previous spawn/sibling particle. These data can be used to establish chains of sibling spawns. The data channel is available only if the Total Spawn Count and Last Spawn ID channels are defined. The data are continuous through integration steps; therefore, if a particle has spawns during several integration steps, you can still establish the full chain of sibling spawns for the parent particle.

Current Spawn Count – Integer data that indicates how many particles were spawned per parent particle during the current integration step.

Total Spawn Count – Integer data that indicates the total amount of particle spawns for a parent particle.

Current Spawn Order – Integer data that indicates the order of spawns in the current integration steps, from 0 to N-1, where N is number of spawns for a parent particles in the current integration step. The data can be used to adjust particle location in space, for example to spread particles from the location of the last spawn particle in the previous integration step to the current location of the parent, thus creating placement similar to the standard Spawn operator.

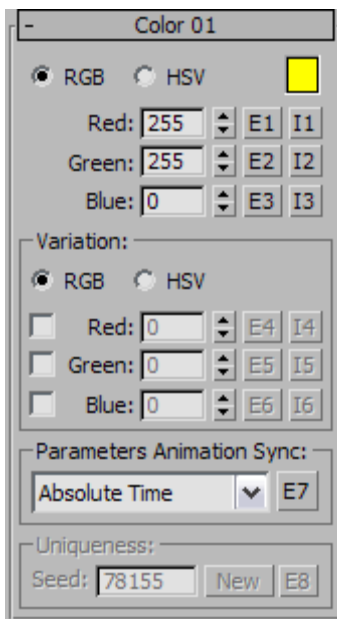
Color Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow. > Add or select Color.

The Color suboperator augments the usage of the [Vector suboperator](#). As with Vector, Color generates Vector-type data output. However, the parameters of the Color suboperator are more convenient if you plan to use vector data as vertex-color mapping for particles.

For a usage example, see the included file *ColorSuboperator.max*: Render frame 30 to see the effect of the suboperator usage.

Interface



RGB/HSV – Determines whether color information is set up in RGB or HSV form. Keep in mind that even if RGB or HSV values are set as integers in the range [0,255], the suboperator's output is vector values with real components in the range from 0.0 to 1.0. This is the form suitable for vertex color information.

Color Swatch – Click to set RGB/HSV values via a Color Picker dialog.

Red/Green/Blue or Hue/Sat/Val – These are the color components, and can be animated.

Each component has complementary buttons in the form E# or I#. The E# button creates an [Equal-type](#) parameter input that you can wire with a Parameter suboperator. This way

you can wire the same value to different suboperators (a single parameter suboperator is wired to several suboperators with E-type input). The I# button defines an integer data input, which lets you wire individual color component values to every particle. Only one type—I# or E#—can be active at a time. When active, it creates an input of the corresponding type.

Variation group – You can define color variation in either RGB or HSV space for the generated colors. When you use variation, the Uniqueness parameters become available.

When you use variation, it's advisable to set a particle's colors only once: when it enters the event, using the [Input Standard](#) suboperator with the New In Event option. Otherwise, a different color is generated at every frame for each particle, which creates a color-blinking effect. The Variation value defines the maximum possible offset from the main color value. The actual offset magnitude is a random value that does not exceed the defined parameter.

The Parameters Animation Sync and Uniqueness groups are similar in functionality to the groups of the same names in Vector suboperator.

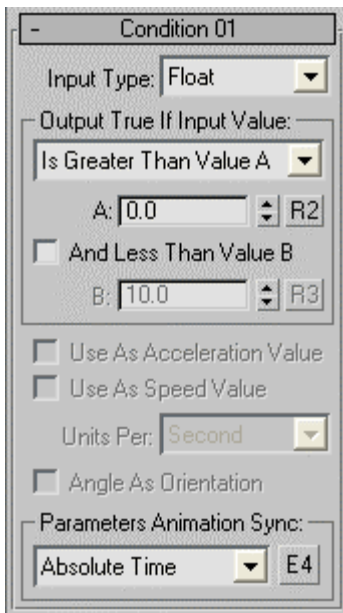
Condition Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Condition.

The Condition suboperator converts input scalar data to a Boolean value (True or False) by comparing it to a single value or testing for whether it falls within a range. You can specify either or both compared-to values explicitly or as inputs from other suboperators.

Note: When you wire the output of a suboperator to the filter input of an Input-type suboperator, a Condition suboperator of the appropriate type is automatically created and placed between the two.

Interface



Input Type – Lets you choose the type of data to be compared. The choices and data types are: Angle (real), Float (real), Integer (integer), Percent (real), Time (time), World Unit (real). This setting determines the type of data output that can be wired to the Condition suboperator input.

Output True If Input Value – This is the primary comparison; it can work on its own or in conjunction with the second (range) comparison, which refers to Value B. Choose the comparison from the drop-down list. For all input types except Integer, there are two choices:

- Is Less Than Value A
- Is Greater Than Value A

The Integer input type also offers these two choices, plus choices for equality and inequality. When the choice is "Less Than" or "Greater Than," the secondary comparison is Greater Than or Less Than, respectively, with respect to Value B.

A – Lets you set an explicit value for the primary comparison.

R2/I2/T2 – Lets you specify input from another suboperator to be used as Value A. When on, adds an input of the appropriate type to the suboperator, to which you can connect any suboperator that outputs the same value. This input value replaces the explicit A value.

And Less/Greater Than Value B – When on, the comparison is based on a range from Value A to Value B. Either value can be the high end or the low end of the range.

B – Lets you set an explicit value for the secondary (range) comparison.

R3/I3/T3 – Lets you specify input from another suboperator to be used as Value B. When on, adds an input of the appropriate type to the suboperator, to which you can connect any suboperator that outputs the same value. This input value replaces the explicit B value.

Use As Acceleration Value/Speed Value – When Input Type=World Unit, you can choose either of these to cause Particle Flow to compare the input value as an acceleration or speed rate, in units per frame, second, or tick. You can activate only Acceleration Value or Speed Value, not both; clicking again turns the option off.

Use As Spin Rate – When Input Type=Angle and this check box is on, Particle Flow compares the input value as a spin rate, in units per frame, second, or tick.

Units Per – Sets the time frame for the acceleration, speed, or spin rate value.

Angle As Orientation – When on, the input value is regarded as an absolute orientation, rather than as a relative rotation. Available only when Input Type=Angle.

Parameters Animation Sync – If you animate the suboperator parameters, the software can begin applying this animation to all particles as of the start frame of the animation or the first frame of the current event, or to each particle based on its age. The options are:

- **Absolute Time** – Any keys set for parameters are applied at the actual frames for which they're set.
- **Event Duration** – Any keys set for parameters are applied to each particle relative to the frame at which it first enters the event.
- **Particle Age** – Any keys set for parameters are applied at the corresponding frames of each particle's existence.
- **Particle Lifespan** – Scales/maps the animation of the parameters onto the particle lifespan period. For example, if a parameter value is animated from 5-20 over

frames 0-100, then this parameter has the value 5 when the particle is born, and the value 20 when the particle dies. This way you can, for example, define the change in a particle's scale over its lifespan.

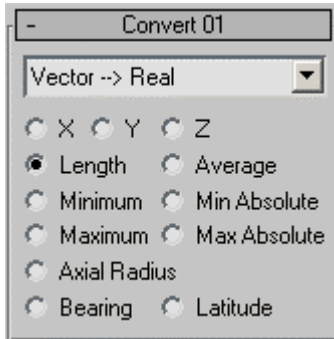
For this option to work properly, there must be a Delete operator set to By Particle Age in the flow to define particle lifespan.

- **Input Proxy** – Adds a Time input to the suboperator, to which you can link any other suboperator that outputs data in Time format.

E4 – Adds an [Equal-type](#) data input for controlling the Animation Sync value. This can receive input only from a [Parameter suboperator](#) set to Type=Animation Sync.

Convert Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Convert.



Convert performs the straightforward task of converting data between compatible formats. For the most part, each choice in the drop-down list has its own interface, which appears below the list box, although some choices have no interface. In some cases, such as Integer -> Compound Index, the interface is read-only, for informational purposes. In other cases, the interface provides a radio button choice. For example, the Real -> Vector choice lets you choose a floating-point value to a vector with Cartesian, cylindrical polar, or spherical polar coordinates. In such cases, the choices are self-explanatory.

Note: When you wire the output of a suboperator to a non-matching input of a different suboperator, a Convert suboperator of the appropriate type is automatically created and placed between the two.

Discretizator Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Discretizator.

Use Discretizator to make float or integer values more discrete, reducing a wide range of values to a more manageable subset at the cost of a certain amount of accuracy. This helps in potential optimization of the input data for the [Shape Control suboperator](#) but can be used for other purposes as well.

The discretization is based on two values: Base and Step. For each incoming value, the Step value is multiplied by an positive or negative integer and then added to the Base value in order to bring the result as close as possible to the input; the input is then rounded to this value. With Base "b" and Step "s," the possible output values from the suboperator is b, b+s, b-s, b+2s, b-2s, b+3s, b-3s, etc.

The incoming values are rounded to the closest discrete value. If incoming value is exactly between the discrete values, it's rounded up to the next-highest value. For example, with Base 0 and Step 1, the input value 1.5 is output as 2, and input value -1.5 becomes -1.

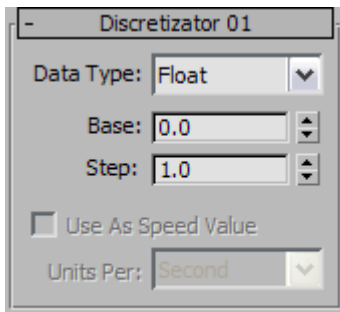
You can find an example of Discretizator usage in the included file *GridAndShapeControl.max*. The flow contains more than 10,000 particles and incorporates two Data operators, both disabled at the beginning.

The first Data Operator does not use Discretizator; turn it on and note how long it takes to calculates the geometry shapes for all particles. It's pretty slow, even on a fast PC.

Next, turn off the first Data operator and turn on the second one. It uses Discretizator in the data flow, and it works much faster when generating particle shapes. This is because the operator makes the incoming data stream discrete with the step of 1 degree. In the first case, the shape is calculated individually for each particle. In the second case the set of possible values is significantly reduced because values are snapped to whole degrees. Because Discrete Optimization is on in the Shape Control suboperator in the second Data Operator, the shapes are generated for 360 particles at most; this is the greatest possible number of different angle values with the step of 1 degree, and then these shapes are shared between particles. The eye cannot distinguish between these two cases, but the second one works much faster.

This, the Discretizator suboperator is a useful optimization tool for the Shape Control suboperator.

Interface



Data Type – The expected incoming data type. This setting determines the number system available for the Base and Step parameters.

Base – The “starting point” for the discretization process. For details, see the introduction to this topic.

Step – This value is added to or subtracted from the Base value repeatedly until the result is as close as possible to the input value. For details, see the introduction to this topic.

Use As Speed Value – Internally, the speed of particles in a Particle Flow system is presented in units per tick (4800 ticks = 1 sec). However, the speed parameters in Particle Flow operators are presented as units per second. To translate from the displayed values to internal values, to turn On Use As Speed Value and use the default setting Units Per Second. This way you can access, for example, the speed defined by the standard Speed operator in Particle Flow. Available only when the input parameter is of the Real type.

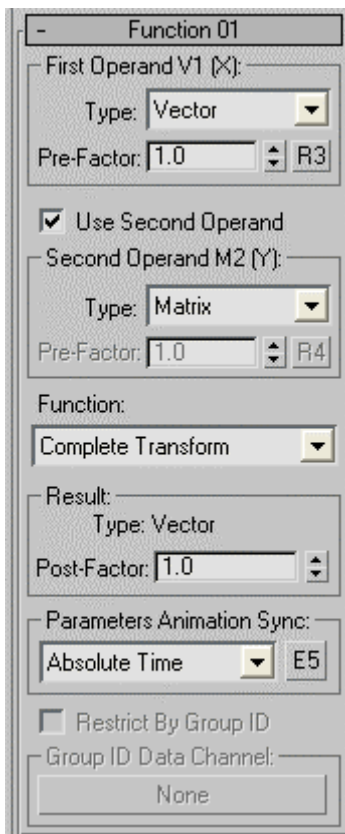
Function Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Function.

The Function suboperator performs one or more mathematical calculations on one or two input values, or *operands*. The suboperator provides a wide variety of functions; the specific available calculations depend on the input type or types.

Note: The output type of the Function suboperator is the more complex of the two operands. For example, a function that uses a real and an integer operand outputs real data. If you need to output a different data type, wire a Convert suboperator to the Function suboperator output.

Interface



First Operand group

The first operand is always available. The group box label ("First Operand") is followed by the label of the respective suboperator input; for example, as shown in the illustration above, it's vector input V1, and then "(X)". The latter means that the input value is substituted for X in the chosen function. For example, if Type=Integer, Use Second

Operand is off, and you use the default function (in this case) of Square X^2 , the input value is used as the first operand and the Function suboperator outputs its square.

Type – Choose the data type of the first operand. All data types supported by Particle Flow Tools: Box #3 are available from the drop-down list except for Complex, Object, Pair, and Equal. This choice determines the group box label and the respective suboperator input connection type.

Pre-Factor – Specifies a real value by which the first operand is multiplied before the function is applied. Default=1.0.

In certain circumstances, the Pre-Factor parameter switches to one of two other parameters:

- **Mix-Factor** – Available only when the Interpolation function is chosen. The Mix-Factor defines the ratio of interpolation of the two operands. For example, if the Mix-Factor is 0.5, then both inputs are considered equally. When interpolating, the sum of shares is 1.0. The Mix-Factor is the share for the first input. Since the sum is 1.0 then the share for the second input is $1.0 - \text{Mix-Factor}$ value.
- **Offset** – Available only when using a single operand (First Operand) of type Integer or Real with the Identity function. The Offset parameter is a quick way to increase or decrease the value in a data channel by a fixed amount without the hassle of creating additional suboperators.

R3 – When on, adds the R3 input to the suboperator, to which you can connect any suboperator that outputs a real value. This input pre-factor replaces the explicit Pre-Factor value, and is multiplied by the first operand before the function is applied.

Use Second Operand – When on, the function uses two operands, with the result that a different set of functions is available on the Function drop-down list. When off, only the First Operand input is used.

Second Operand group

These controls are available only when Use Second Operand (see above) is on. In this circumstance, the Function suboperator performs calculations on two input values. Depending on the first operand type, the second operand may or may not be chosen by the user. For example, if the first operand is of the Time type, the second operand must also be of the Time type. The suboperator input connections are configured accordingly.

The group box label ("Second Operand") is followed by the label of the respective suboperator input; for example, as shown in the illustration above, it's matrix input M1, and then "(Y)". The latter means that the input value is substituted for Y in the chosen

function. For example, if both input types are Time and you use the function Addition X+Y, the T2 input value is added to the T1 input value.

Type – Choose the data type of the second operand, when appropriate. The drop-down list is available only when the first operand type is Integer, Quaternion, Real, or Vector. Otherwise, the software sets the same data type as the first operand and Type is a read-only field. The data types available from the drop-down list depend on the first operand type. For example, if the first operand type is Integer, the second can be only Integer or Real.

Pre-Factor – Specifies a real value by which the second operand is multiplied before the function is applied. Default=1.0.

R4 – When on, adds the R4 input to the suboperator, to which you can connect any suboperator that outputs a real value. This input pre-factor replaces the explicit Pre-Factor value, and is multiplied by the second operand before the function is applied.

Function – Choose the formula used to calculate the base output value. The drop-down list contents depend on the first operand data type and second operand data type, if present. The functions are self-explanatory.

Result group

Type – The output data type of the Function suboperator. With a single operand and with two operands of the same type, this is always the same as the first operand data type. With two operands of different types, the output type is the more complex of the two operands. For example, a function that uses a real and an integer operand outputs real data. If you need to output a different data type, wire a Convert suboperator to the Function suboperator output.

Post-Factor – Specifies a value of the result type by which the result is multiplied after the function is applied. Default=1.0 or 1.

Parameters Animation Sync – If you animate the suboperator parameters, the software can begin applying this animation to all particles as of the start frame of the animation or the first frame of the current event, or to each particle based on its age. The options are:

- **Absolute Time** – Any keys set for parameters are applied at the actual frames for which they're set.
- **Event Duration** – Any keys set for parameters are applied to each particle relative to the frame at which it first enters the event.
- **Particle Age** – Any keys set for parameters are applied at the corresponding frames of each particle's existence.

- **Particle Lifespan** – Scales/maps the animation of the parameters onto the particle lifespan period. For example, if a parameter value is animated from 5-20 over frames 0-100, then this parameter has the value 5 when the particle is born, and the value 20 when the particle dies. This way you can, for example, define the change in a particle's scale over its lifespan.

For this option to work properly, the flow must contain a Delete operator set to By Particle Age to define particle lifespan.

- **Input Proxy** – Adds a Time input to the suboperator, to which you can link any other suboperator that outputs data in Time format.

E5 – When on, you can expose the animation sync parameter via a [Parameter suboperator](#) and let the user choose. Turn on E5, add a Parameter suboperator set to Type: Animation Sync, wire it to the E5 input on the Function suboperator, and then use [Expose Parameters](#) to make the setting available in the Particle View interface.

Restrict By Group ID – When on, you can specify a Group ID integer data channel on which to execute the Function suboperator. Specify the data channel by clicking the Group ID Data Channel button.

This option is available only under the following conditions:

- For "All"-type functions, which appear in the Function drop-down list when Use Second Operand is off.
- For "Average"-type functions, which can appear in the Function drop-down list when using either one or two operands.

When Restrict By Group ID is on and you've set a group ID, the specified function is executed on all particles with that group ID, but no others.

Here's an example of how this option can work:

Particle data=1 2 3 4 5 1 2 3 4 5

Function=Average: Result=3 3 3 3 3 3 3 3 3 3

Now, suppose Restrict By Group ID is on, and the data in the specified Group ID Data Channel setting=1 1 1 2 4 4 4 4 8

The group ID restricts how the average value will be calculated - it is for the particles with the same group ID. So, the function result will be as following (from the original particle data): 2 2 2 4 3 3 3 3 5

The original example had the following use case. The particles are given group IDs according to their parents: If they are spawned from the same parent then they have the

same group ID. In this case, particles travel and then collide with a mesh. Once a particle collides with a mesh, *all* particles with the same group ID should jump to the next event.

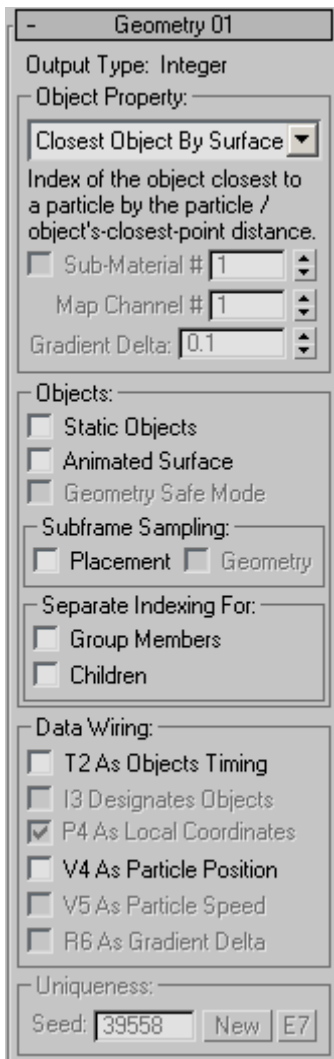
This can be accomplished in this way: Once a particle collides, its data changes from False to True. Then OR / All Particles is used with Restrict By Group ID on. Therefore, by using OR on the same group, the True value propagates to all particles in the same group.

Geometry Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Geometry.

The Geometry suboperator provides access to geometric and related data in the 3ds Max scene. Input is of the Object data type, and typically comes from the output of a Select Object suboperator.

Interface



Output Type – Displays the type of data output by the Geometry suboperator; this is determined by the Object Property choice.

Note: The Pair and Complex output types for this suboperator, related to certain choices here, are discussed in [Data Types](#).

Object Property – Choose the type of property from the drop-down list. A brief explanation of the active property appears immediately below the list box.

Sub-Material – Turn on to access a specific member of a multi-material, such as Multi/Sub-Object. Use the numeric field to specify the sub-material value. Available only with the following object properties: Point Color, Point Color 3D, Point Color Gradient, Point Color Gradient 3D, Point Self-Illumination, Point Opacity,

Map Channel – Lets you specify a UVW mapping channel to access. Available only with the Point Mapping and Point Mapping Gradient properties.

Gradient Delta – Defines a distance in world space. This parameter is used for the option where a gradient is calculated by querying properties in the vicinity of a central point. The parameter defines the size of this vicinity. Available only with the Point Color Gradient, Point Color Gradient 3D, and Point Mapping Gradient properties.

Objects group

Static Objects – Turn on if the reference objects do not move, rotate, or scale; that is, their transformation matrix is constant. This speeds up object evaluation.

Animated Surface – Turn on if the surface of the reference objects is animated via modifiers, object parameters, or sub-object animation.

Geometry Safe Mode – Turn this on if the reference surface is a regular manifold surface, that is, every edge has one or two faces coming out of it. This speeds up the construction of binary search trees for geometry evaluation.

Subframe Sampling group

Placement – By default this option is off, and the transformation matrix of the object has accuracy of one frame. When on, the transformation matrix is queried on a sub-frame basis, for greater accuracy at the cost of slower computation.

Geometry – By default this option is off, causing the shape of the object to be considered static within the current frame. When on, the geometry/shape of the object is queried on sub-frame basis, resulting in greater accuracy at the cost of slower computation.

Separate Indexing For group

Group Members – When on, group members are considered as separate objects. When off, the entire group is treated as a single object.

Children – When on, hierarchy members are considered as separate objects. When off, the entire hierarchy is treated as a single object.

Data Wiring group

T2 As Objects Timing – Adds the T2 input connector to the suboperator in Data View; any other suboperator that outputs the Time data type can be wired to this. The Time data input defines the exact timing to get geometry properties from the reference objects. Keep in mind that if the input timing is not in sync for all particles, the Geometry suboperator may take on a significant workload, since it would need to determine unique object states (as defined by different timing) for each particle. Available with all Object Property settings.

I3 Designates Objects / Designates Objects/Verts – Adds the I3 input connector to the suboperator in Data View; any other suboperator that outputs the Integer data type can be wired to this. When off, the property is calculated as considered for all objects. For example, Closest Point means the closest point from *all* reference objects. When on, the property is calculated for a specific reference object (the index supplied is the index in the list of the reference objects in the [Select Object suboperator](#)).

Available only with Object Property choices that make sense for designating particular objects: Closest Point, Closest Point Distance, Closest Vertex, Collision Point, Inside Objects, Object Number of Faces/Vertices, Object Size, Object Surface, Object Volume, Point Color 3D, and Point Color Gradient 3D.

With Vertex ... properties such as Vertex Illum, the option is on by default but is unavailable for turning off, and is named I3 Designates Objects/Verts, because for these properties the I3 input defines the objects/vertices indices to get properties from. For Vertex ... properties this input is a Compound Index (object index / vertex index); for all other options (where the default state is off), it's just an object index.

P4 As Local Coordinates – Adds the P4 input connector to the suboperator in Data View; any other suboperator that outputs the [Pair data type](#) can be wired to this.

The Face Area and Face Selection options can be used to calculate the area of the face of an object or the selection status of a face. And although it seem to be sufficient to supply a compound index (object index + face index) as an index, the suboperator has a Pair-type input for these two options. This is done to simplify the wiring from the Closest Point option that has Pair as an output. If the object index + face index are created in a different way then you can use a Convert suboperator to create a Pair type – just use a zero vector as the other component of the Pair type.

V4 As Particle Position – Adds the V4 input connector to the suboperator in Data View; any other suboperator that outputs the Vector data type can be wired to this. Usually, geometry properties such as Closest Point are calculated with regard to the current particle position. However, when this option is available, you can define the Vector data channel to be considered as the "particle position," that is, the location to be considered for calculating the geometry property.

V5 As Particle Speed – Adds the V5 input connector to the suboperator in Data View; any other suboperator that outputs the Vector data type can be wired to this. This option is available for the Collision Point property only. The collision point is calculated from the knowledge of the current particle speed. However, if you would like to calculate the possible collision in the future (for avoidance purposes), you might want to modify the speed vector/value to some artificial value; for example, to increase the speed value to foresee a possible future collision. To do so, you can modify current particle speed by using Input Standard and Function suboperators, and wire it into the V5 input.

R6 As Gradient Delta – Adds the R6 input connector to the suboperator in Data View; any other suboperator that outputs the Real data type can be wired to this. Available only with the "... Gradient" properties.

Uniqueness group

Seed – Lets you set a random seed to differentiate behavior from other randomly seeded functions. Change the Seed value to vary the random behavior, either by entering a value or clicking New. Available only with the Random Surface Point and Random Volume Point object properties.

E7 – When on, you can expose the Seed parameter via a [Parameter suboperator](#) and let the user choose. Turn on E7, add a Parameter suboperator set to Type: Uniqueness Seed, wire it to the E7 input on the Geometry suboperator, and then use [Expose Parameters](#) to make the setting available in the Particle View interface.

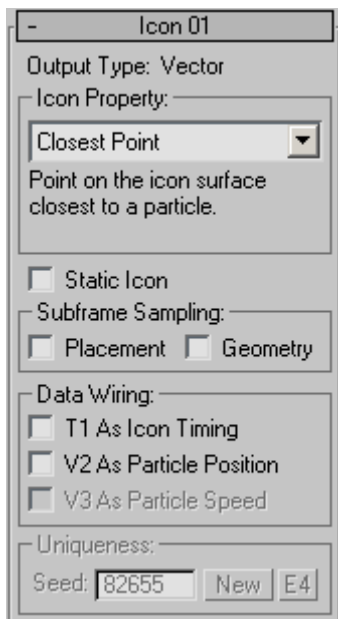
Icon Suboperator

Path: Particle View > Data Icon/Data Icon Test > Click Edit Data Flow > Add or select Icon.

The Icon suboperator is available only with the Data Icon operator and the Data Icon Test. It lets you determine properties of the icon associated with the operator/test for use in calculations in the operator/test.

When you add a Data Icon operator or Data Icon Test to the particle system, Particle Flow creates an icon and places it at the 3ds Max world origin. You can control the icon's physical properties in Particle View via the rollout for the operator/test. You can perform calculations within the operator/test based on a specific property of the icon with the Icon suboperator. For example, you can use the Icon suboperator to measure the distance between a particle and the icon, or simply determine the speed or angular velocity of the icon.

Interface



Output Type – Indicates the data type the suboperator outputs. This is determined by the Icon Property choice.

Icon Property – Choose the property type for the icon from the drop-down list. A brief description of the chosen property appears directly below the list box.

Static Icon – Turn on if the icon does not move, rotate, or scale; that is, its transformation matrix is constant. This speeds up object evaluation.

Subframe Sampling group

Placement – By default this option is off, and the transformation matrix of the icon has accuracy of one frame. When on, the transformation matrix is queried on a sub-frame basis, for greater accuracy at the cost of slower computation.

Geometry – By default this option is off, causing the shape of the icon to be considered static within the current frame. When on, the geometry/shape of the icon is queried on sub-frame basis, resulting in greater accuracy at the cost of slower computation.

Data Wiring group

T1 As Icon Timing – Adds the T1 input connector to the suboperator in Data View; any other suboperator that outputs the Time data type can be wired to this.

V2 As Particle Position/Local Coordinates – Adds the V2 input connector to the suboperator in Data View; any other suboperator that outputs the Vector data type can be wired to this. The V2 As Particle Position option is available only for property types for which it's necessary to know the particle position, such as Distance To Icon. With property types for which it's necessary to know local coordinates, such as Point Normal, the option is labeled V2 As Local Coordinates and is on but unavailable; it can't be turned off.

V3 As Particle Speed – Adds the V3 input connector to the suboperator in Data View; any other suboperator that outputs the Vector data type can be wired to this. This option is available only with the Collision Point property.

Uniqueness group

Seed – Lets you set a random seed to differentiate behavior from other randomly seeded functions. Change the Seed value to vary the random behavior, either by entering a value or clicking New. Available only with the Random Surface Point and Random Volume Point icon properties.

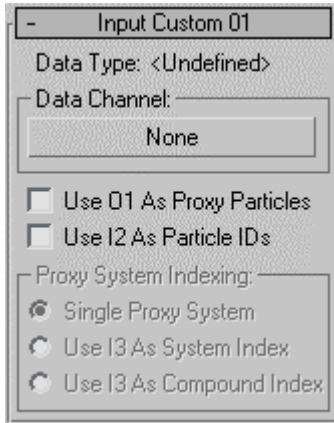
E4 – When on, you can expose the Seed parameter via a [Parameter suboperator](#) and let the user choose. Turn on E4, add a Parameter suboperator set to Type: Uniqueness Seed, wire it to the E4 input on the Icon suboperator, and then use [Expose Parameters](#) to make the setting available in the Particle View interface.

Input Custom Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Input Custom.

Input Custom copies data from a channel created by the [Output New suboperator](#).

Interface



Data Type – This read-only field shows the data type of the Input Custom suboperator, as defined by the specified Output New suboperator. When no suboperator is specified, the data type is undefined.

Data Channel – The Output New suboperator used by the Input Custom suboperator. Click the button and then choose an Output New suboperator from the Select Data Channel dialog. The suboperator's name then appears on the button.

Use O1 As Proxy Particles – When on, adds an Object-data input to the Particles suboperator. You can connect a particle flow from the Select Object suboperator to this input. The proxy particles are used for property aggregation. Default=on.

Use I2 As Particle IDs – When on, adds an Integer-data input to the Particles suboperator for inputting particle-ID data.

This option lets you remix the custom particle data. For example, if the input integer data is { 2, 4, 6, 8, 10, etc. } then the first particle in the current event gets the custom data from a particle with Particle ID = 2; the second particle in the current event get the custom data from a particle with Particle ID = 4, etc.

Priority System Indexing group

See [Priority And Execution Order](#).

Input Proxy Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Input Proxy.

The Input suboperators serve as inputs to your Data operator or test. They bring in particle properties from a particle system and output an appropriate data type for the selected option.

The Input Proxy suboperator lets you bring in particle properties from proxy particle systems – that is, particle flows other than the current one. It always has an O1 (Object data) input to which you wire a Select Object suboperator for specifying the proxy particle system. Additional, optional inputs are I2 for specifying particle IDs and I3 for use as a system or compound index.

Interface

The image shows a software interface for configuring an 'Input Proxy 01'. The interface is organized into several sections:

- Output Type:** Real
- Proxy System Indexing:**
 - Use I2 As Particle IDs
 - Single Proxy System
 - Use I3 As System Index
 - Use I3 As Compound Index
- Accel:** Magnitude (dropdown)
- ID:** Birth Index (dropdown)
- Mapping:** Vector (dropdown)
- Channel #:** 1 (spinner) E4 (text)
- Mass:** (radio button)
- Material Index:** (radio button)
- Position:** Vector (dropdown)
- Rotation:** Euler Angles (dropdown)
- Scale:** Average (dropdown)
- Script:** Float (dropdown)
- Selected By Group:** (radio button)
 - Group Selection Operator:** None (text)
 - Type:** Post-Step Status (dropdown)
- Selected By Particle System:** (radio button)
 - Shape:** # Vertices (dropdown)
 - Adjusted By Scale
 - Size:** Maximum (dropdown)
 - Speed:** Magnitude (dropdown)
 - Spin:** Rate (dropdown)
 - Time:** Particle Age (dropdown)
 - TM:** Matrix (dropdown)
- Vertex Color Channel:** (radio button)

Output Type – This read-only field shows the output type of the suboperator, based on the active choice of the type of data to bring in from the proxy particle system.

Proxy System Indexing group

Use I2 As Particle IDs – Adds an Integer-type input, to which you can wire a suboperator that outputs the data to use as particle IDs.

This option lets you remix the custom particle data. For example, if the input integer data is { 2, 4, 6, 8, 10, etc. } then the first particle in the current event gets the custom data from a particle with Particle ID = 2; the second particle in the current event get the custom data from a particle with Particle ID = 4, etc.

Typically you'll use just one proxy particle system, so you can leave this setting at the default choice:

- **Single Proxy System** – The suboperator references only the input particle system (O1) as a proxy; no further information needed.
- **Use I3 As System Index** – Adds an Integer-type input, to which you can wire a suboperator that outputs the data to use as the ID of the particle system to use as the proxy.
- **Use I3 As Compound Index** – When a Particles suboperator is used to calculate Closest Particle Index, and its Use O1 As Proxy Particles check box is on, then it can calculate the closest particles from several particle systems if the Select Object suboperator that it uses as input has a list of particle systems. As a result, the index of the closest particle should include the index of a particle system and the ID of the closest particle in this particle system. These integer values are compacted into a single *compound index* that can be used later by other suboperators, for example by the Input Proxy suboperator that has Use I3 As Compound Index turned on. When this option is on, it indicates that the Select Object suboperator used as the input has a list of particle systems, and the properties from proxy particles are gathered from several particle systems. The compound index has the information about the particle system index and the particle ID. You can also create the compound index values from two integer values by using the Convert suboperator.

The remaining Input Proxy settings let you choose the property or properties to copy from the proxy particle system.

Accel – Acceleration, which can be expressed as a magnitude value, vector data, or a component (X, Y, or Z) of the vector data.

Note: None of the standard PFlow operators create the Acceleration or Mass property. However, if the Acceleration property is created (using the [Output Standard suboperator](#)), the PFlow system obeys the rules of the supplied acceleration. For example, to simulate the effect of gravity, you can define the acceleration value as an output.

ID – Choose one of the ID types:

- **Birth Index** – The index number assigned to each particle at birth.
- **Event Index** – The index number of the event in which the particle currently resides.
- **Uniform Index** – Each particle has a unique Birth Index value, given to it at birth. You can control the number of particles in a Particle Flow system either with the settings in a Birth-type operator, or by changing the Multiplier settings in PF Source object. Suppose that a Birth operator is set to generate 1,000 particles and the Multiplier value is set to 10%. In this situation only 100 particles are born, and they have Birth Index values of 0 to 99. The Uniform Index stretches uniformly for all particles generated, regardless of the Multiplier value. For our example, the last particle born always has a Uniform Index equal to 999. As a result, when the Multiplier value exceeds 100%, the Uniform indices are not unique.

Typically one uses different Multiplier values for Viewport and Render. The Uniform Index parameter is useful when the overall effect should be independent of the number of particles as affected by the Multiplier values.

Mapping – Lets you use mapping information from the particle system, either as a vector or a U/V/W component. In any case, use the Channel # setting to specify the mapping channel, or turn on E4 and wire a [Parameter suboperator](#) to the E4 input, and then expose the Value parameter in the Parameter suboperator so the user can set the mapping channel.

Mass – Mass as a particle property.

Material Index – The index assigned by the Material operators (Static, Dynamic and Frequency) when they deal with multi/sub-object materials and assign sub-material IDs to particles.

Position – The current position of the particle, in 3D space as Vector data, or the X, Y, or Z component of the position as Real data.

Rotation – The orientation of the particle in one of four data types:

- Angle=Real
- Axis=Vector
- Euler Angles=Vector
- Quaternion

Scale – Offers a number of different methods of determining particle scale. All are Real data except the Vector option.

Script – Lets you read data as defined by the Script operators (Script Operator, Script Test and Birth Script) in the script data channels: Integer, Float, Vector, Matrix.

Selected By Group – This option applies only if you have Toolbox #1 installed. In that case, and if the particle system uses a Group Selection operator, you can determine the selection state and selection time as defined by the Group Selection operator. Click the None button to open a dialog that lets you choose the Group Selection operator to use.

Type – Lets you specify the timing for Selected By Group:

- **Pre-Step Status** – the selection status of particles before the current frame
- **Post-Step Status** – the selection status of particles after the current frame
- **Switch Time** – time when the selection status of particles toggles during the current frame

Selected By Particle System – Outputs a Boolean: Yes/1 for particles that are selected in the particle system, or No/0 for particles that are not.

Shape – Outputs shape data in one of the following categories:

- **Extent** – Adds a Vector (V2) input for defining direction. The output Real value is the shape extent from the pivot point to the shape boundary in the given direction. For example, if a particle shape is a sphere, then the output real value is the radius of the sphere (whatever the input vector value is). If a particle shape is a cube in the canonical position, then for input vector (1, 0, 0) the output value is the half-size of the cube.
 - **Adjusted By Scale** – Defines whether the particle shape property is calculated on the shape geometry information alone (off) or the size of the shape is adjusted as defined by the Scale particle channel (on). Available only when the Shape choice is Extent.
- **# Faces** – Outputs Integer data.
- **# Vertices** – Outputs Integer data.
- **Surface** – Outputs Real data. Enables the Adjusted By Scale check box for the surface of a particle shape.
- **Volume** – Outputs Real data. Enables the Adjusted By Scale check box for the volume of a particle shape.

Size – Outputs size data in one of a number of available categories (self-explanatory). All output Real data except the Vector option.

Speed – Outputs velocity data in one of a number of available categories (self-explanatory). All output Real data except the Vector option.

Spin – Outputs spin data in one of three categories: Axis (Vector), Quaternion, and Rate (Real).

Time – Outputs time data in one of a number of available categories (self-explanatory).

TM – Outputs the particle transformation matrix as Matrix data comprising three vectors, or as Matrix X, Y, or Z (Vector).

Vertex Color Channel – Outputs vertex color channel data as a Vector.

Input Standard Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Input Standard.

The Input suboperators serve as inputs to your Data operator or test. They bring in particle properties from a particle system and output an appropriate data type for the selected option.

Input Standard copies regular particle properties such as position, speed, scale, etc., from the current particle system. The parameters are the same as the [Input Proxy suboperator](#), except that there are no Proxy System Indexing controls. Also, Input Standard allows for an additional property:

Visibility – This is a Boolean data type; a particle is either hidden or visible. The options are:

- **Viewport** – The property is common to all Display operators.
- **Render** – The property is common to all Render operators.
- **By Operator** – The property applies to a specific Display or Render operator. Choose this, and then use the Viewport/Render Operator button to choose the operator.

Bear in mind that each Display and Render operator has its own Visible % parameter. That parameter is independent one, and has nothing to do with the Visibility property from a Data operator. The standard Display and Render operators obey Visibility data control from a Data operator. Display Data and Display Script also obey the Visibility data control; however you need to download the latest version (build 1.063 or higher) of the PFTools: Freebies plug-in (Display Script operator).

Interface

- Input Standard 01

Output Type: Real

Accel: Magnitude ▾

ID: Birth Index ▾

Mapping: Vector ▾

Channel #: 1 ▾ E

Mass

Material Index

New In Event

Position: Vector ▾

Rotation: Euler Angles ▾

Scale: Average ▾

Script: Float ▾

Selected By Group

Group Selection Operator:

None

Type: Post-Step Status ▾

Selected By Particle System

Shape: # Vertices ▾

Adjusted By Scale

Size: Maximum ▾

Speed: Magnitude ▾

Spin: Rate ▾

Time: Particle Age ▾

TM: Matrix ▾

Vertex Color Channel

Visibility: Render ▾

Viewport/Render Operator:

None

Notes Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Notes.

The Notes suboperator, like the Particle Flow operator of the same name, has no effect on the particle system; its purpose is simply for adding comments regarding the entire Data operator/test. Also, as with Particle Flow, you can add comments to individual suboperators by right-clicking them and choosing Comments from the context menu.

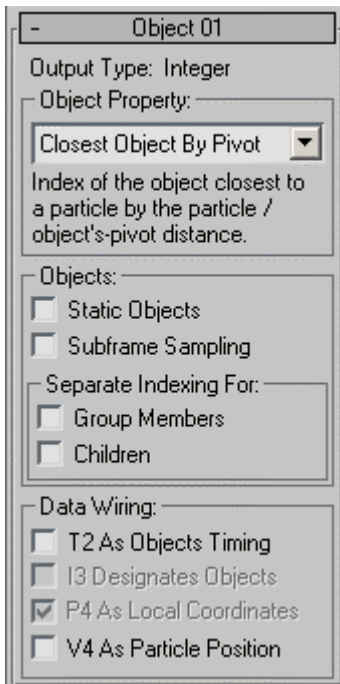


Object Suboperator

Path: Particle View > Data Icon/Data Icon Test > Click Edit Data Flow > Add or select Object.

Use the Object suboperator to acquire properties from reference objects.

Interface



Output Type – This read-only field displays the suboperator output type as determined by the active Object Property setting.

Object Property – Lets you choose the property to acquire from a drop-down list. A brief explanation of the active choice appears under the list box.

Note: For an explanation of the Pair input used by the Point Position option, see [Data Types](#).

Objects group

Static Objects – When on, indicates that the object's shape is not animated.

Subframe Sampling – When on, the data is sampled more frequently; the benefit is greater accuracy, while the cost is longer processing times.

Separate Indexing For – Turn on Group Members or Children to account for individual objects within groups or hierarchies.

Data Wiring group

T2 As Objects Timing – When off, the object’s properties are acquired at the current frame. When on and Time channel data are plugged in, the object’s timing can be adjusted according to the input Time data. This way it is possible to get the Object pivot position, say, 10 frames ahead—you can use a Value suboperator that has an animated Time output that runs 10 frames ahead of the current frame, and wire it into the Object suboperator.

I3 Designates Objects – Adds an Integer input for specifying an object index as defined by the [Select Object suboperator](#).

P4 As Local Coordinates – Activates a Pair input for specifying local coordinates.

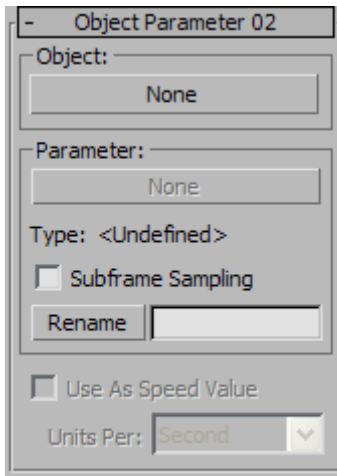
V4 As Particle Position – Activates a Pair input for specifying local coordinates.

Object Parameter Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Object Parameter.

Use Object Parameter to get a single parameter value from any object in the scene, including modifier settings and Particle Flow operators and tests.

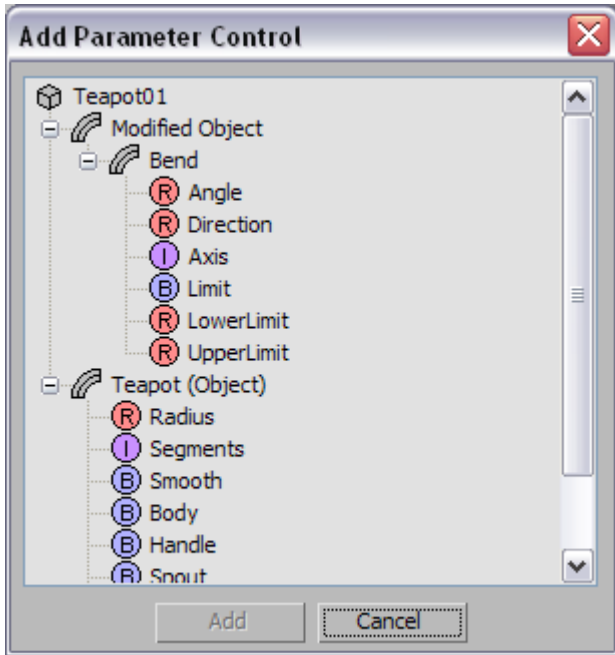
Interface



Object – The object from which to derive a parameter. Click this button and then select the object whose parameter you want to use. This adds the object to the suboperator and displays its name on the Object button. Use the Parameter setting (see following) to specify the parameter.

To use a parameter from a Particle Flow element, select the system icon, such as PF Source 01.

Parameter – The object parameter for the suboperator to use. After specifying the object (see preceding), click this button to open the Add Parameter Control dialog. Choose the parameter from the list in the dialog and click Add.



Type – The data type for the parameter, such as Real or Boolean.

Subframe Sampling – When on, provides extra precision with an animated parameter whose value is changing quickly at the subframe level; this is particularly useful when particles come to the event at slightly different times. When Off, the parameter value is queried as of the end of the integration step.

Usually it's not necessary to turn on this setting even if the parameter is animated. It helps primarily in the specific conditions described above.

Rename – To change the parameter name, enter a new name and click Rename. This is useful for giving the parameter a more meaningful name in the context of the suboperator.

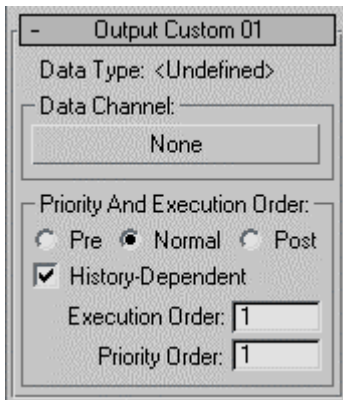
Use As Speed Value – Internally, the speed of particles in a Particle Flow system is presented in units per tick (4800 ticks = 1 sec). However, the speed parameters in Particle Flow operators are presented as units per second. To translate from the displayed values to internal values, to turn On Use As Speed Value and use the default setting Units Per Second. This way you can access, for example, the speed defined by the standard Speed operator in Particle Flow. Available only when the input parameter is of the Real type.

Output Custom Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Output Custom.

The Output Custom suboperator transfers calculated data into a data channel created by the [Output New suboperator](#).

Interface



Data Type – This read-only field displays the data type of the channel specified by the Data Channel control (see following).

Data Channel – Click this button and then use the dialog that opens to choose the Output New suboperator from which to transfer.

Priority And Execution Order group

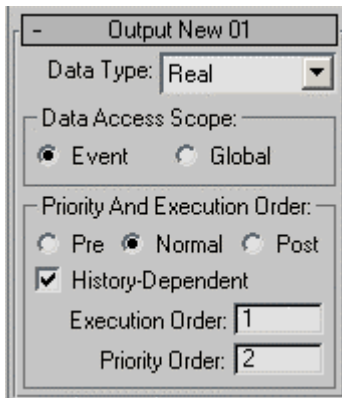
See [Priority And Execution Order](#).

Output New Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Output New.

The Output New suboperator creates a new data channel that can be used by the Input Custom and Output Custom suboperators.

Interface



Data Type – Choose the output data type. Most of the data types are self explanatory; for a discussion of the more complex types, see [Data Types](#).

Note: Thanks to the fact that Output New supports the Object data type, you can use the same reference objects in several Data operators, and you need to set them up in only one Data Operator. To do this, wire the output of a [Select Object](#) suboperator to an Output New suboperator, and then use an [Input Custom](#) suboperator in a different Data operator to extract the object reference selection that was done in the first Data Operator.

Data Access Scope – Choose whether the suboperator should work within the current event only (Event), or throughout the particle system (Global).

Priority And Execution Order group

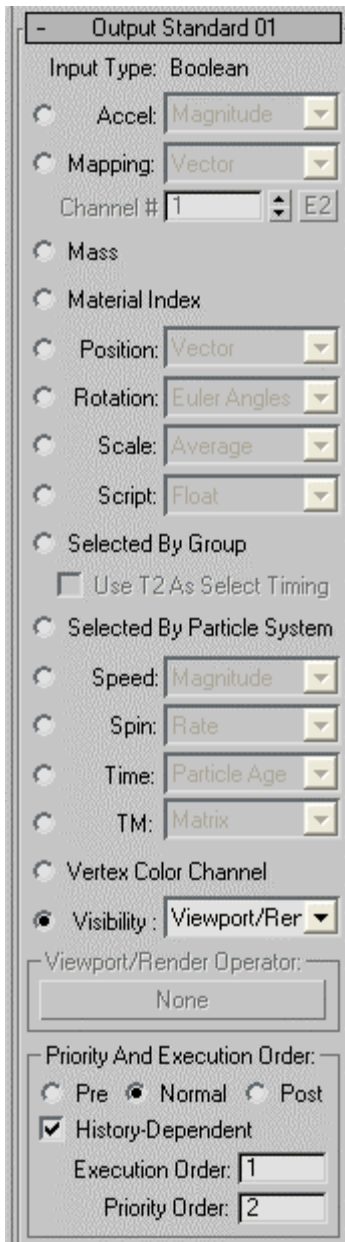
See [Priority And Execution Order](#).

Output Standard Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Output Standard.

The Output Standard suboperator transfers calculated data into regular particle properties. To use it, choose an input type, and then wire a suboperator that outputs that data type to the input connector on the suboperator in Data View. Output Standard then sends that data out into the particle system to which the Data operator/test belongs.

Interface



Input Type – This read-only field displays the data type the suboperator accepts, as determined by the chosen data type. The choices are:

Accel – Acceleration, which can be expressed as a magnitude value, vector data, or a component (X, Y, or Z) of the vector data.

Note: None of the standard PFlow operators create the Acceleration or Mass property. However, if the Acceleration property is created (using the [Output Standard suboperator](#)), the PFlow system obeys the rules of the supplied acceleration. For example, to simulate the effect of gravity, you can define the acceleration value as an output.

Mapping – Outputs mapping information as Vector data, accounting for mapping in all three dimensions, or as Real data for mapping in a single dimension U, V, or W. With this option chosen, you can set the mapping channel to which the data is applied (1-99), or click E and wire a Parameter suboperator to the E2 input so that the user can specify the channel.

Mass – Mass as a particle property.

Material Index – Accepts an Integer input, which it outputs as a material index.

Position – The current position of the particle, in 3D space as Vector data, or the X, Y, or Z component of the position as Real data.

Rotation – The orientation of the particle in one of four data types:

- Angle=Real
- Axis=Vector
- Euler Angles=Vector
- Quaternion

Scale – Offers a number of different methods of outputting particle scale data. All are Real data except the Vector option.

Script – Writes the calculated data into script channel data: the same channel data as used by the Script operators (Script Operator, Script Test and Birth Script).

Selected By Group – This option applies only if you have Toolbox #1 installed. In that case, and if the particle system uses a Group Selection operator, you can determine the selection state and selection time as defined by the Group Selection operator. Click the None button to open a dialog that lets you choose the Group Selection operator to use.

Use T2 As Select Timing – The suboperator adjusts the selection status data in a Group Selection operator (from PFTools:Box#1). When on, you can use the T2 input to define when the selection status toggles (from True to False, or from False to True). When off, it

is presumed that the selection status data timing is as the beginning of the current frame. Box#1 can use the exact timing of the selection data toggle for some effects.

Selected By Particle System – Accepts a Boolean value: Yes/1 for particles that are selected in the particle system, or No/0 for particles that are not.

Speed – Inputs velocity data in one of a number of available categories (self-explanatory). All require Real data except the Vector option.

Spin – Inputs spin data in one of three categories: Axis (Vector), Quaternion, and Rate (Real).

TM – Inputs the particle transformation matrix as Matrix data comprising three vectors, or as Matrix X, Y, or Z (Vector).

Vertex Color Channel – Inputs vertex color channel data as a Vector.

Visibility – This is a Boolean data type; a particle is either hidden or visible. The options are:

- **Viewport** – The property is common to all Display operators.
- **Render** – The property is common to all Render operators.
- **By Operator** – The property applies to a specific Display or Render operator. Choose this, and then use the Viewport/Render Operator button to choose the operator.
- **Viewport/Render** – The property is common to all Display and Render operators.

Bear in mind that each Display and Render operator has its own Visible % parameter. That parameter is independent one, and has nothing to do with the Visibility property from a Data operator. The standard Display and Render operators obey Visibility data control from a Data operator. Display Data and Display Script also obey the Visibility data control; however you need to download the latest version (build 1.063 or higher) of the *Particle Flow Tools: Freebies* plugin (Display Script operator).

Priority And Execution Order group

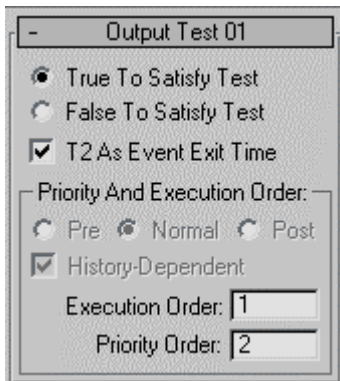
See [Priority And Execution Order](#).

Output Test Suboperator

Path: Particle View > Data Icon Test/Data Test > Click Edit Data Flow > Add or select Output Test.

The Output Test suboperator serves as the output when the Data Icon Test or Data Test is to be used for decision making. It simply lets you specify whether the Boolean input must be true or false to satisfy the test. Optionally, you can also use a Time input to specify when particles should exit the event.

Interface



True/False To Satisfy Test – This read-only field displays the data type the suboperator accepts, as determined by the chosen data type. The choices are:

T2 As Event Exit Time – The input Time channel defines when a particle satisfies the test (since it's a part of the Data Test or Data Icon Test) in the current frame.

Priority And Execution Order group

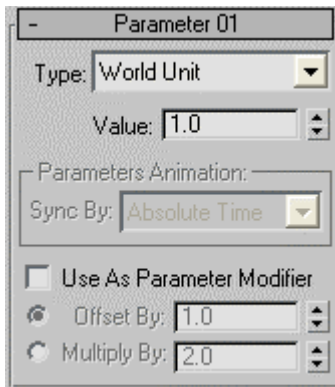
See [Priority And Execution Order](#).

Parameter Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add Or Select Parameter.

The Parameter suboperator defines exposable values for other suboperators. It is particularly useful when you want to expose the same value for two more different suboperators, such as a random seed, for the purpose of keeping your operator interface simple. For more information, see [Equal Data Type](#).

Interface



Type – Choose the data type for the parameter from the list:

- Angle
- Animation Sync
- Float
- Integer
- Percent
- Time
- Uniqueness Seed
- World Unit

With most data types, you can set only the default value; exposing the parameter shows this value in the particle view UI and lets the user change it.

Value – Specifies the default value for the parameter.

Parameters Animation/Uniqueness – Available with the Animation Sync and Uniqueness Seed data types, respectively. With Animation Sync, choose a Sync By value from the drop-down list; with Uniqueness Seed, set a Seed value.

Sync By – If you animate the suboperator parameters, the software can begin applying this animation to all particles as of the start frame of the animation or the first frame of the current event, or to each particle based on its age. The options are:

- **Absolute Time** – Any keys set for parameters are applied at the actual frames for which they're set.
- **Event Duration** – Any keys set for parameters are applied to each particle relative to the frame at which it first enters the event.
- **Particle Age** – Any keys set for parameters are applied at the corresponding frames of each particle's existence.
- **Particle Lifespan** – Scales/maps the animation of the parameters onto the particle lifespan period. For example, if a parameter value is animated from 5-20 over frames 0-100, then this parameter has the value 5 when the particle is born, and the value 20 when the particle dies. This way you can, for example, define the change in a particle's scale over its lifespan.

For this option to work properly, there must be a Delete operator set to By Particle Age in the flow to define particle lifespan.

Use As Parameter Modifier – When on, modifies an existing parameter, either by offset, or multiplication factor.

Using the same random seed in several different suboperators can cause correlation issues between different properties. In this case you can create a Parameter suboperator with Type set to Uniqueness Seed, and feed it into one suboperator. For another suboperator, create an additional Parameter suboperator, turn on Use As Parameter Modifier option, and set it to the same data type. Then the parameter connection should go *through* this additional Parameter suboperator. This way the random seed has an offset, and the random values in the underlying suboperators are not correlated. At the same time you can use a single (original) Parameter suboperator for parameter exposure.

For a usage example, see the included sample file *RandomWalk.max*. To view the underlying logic of the Random Walk operator, add a Data operator and then use Load Preset to open the Random Walk preset.

Offset By/Multiply By – Choose a method and set an amount for modifying the parameter. For the Uniqueness Seed type, only the Offset By option is available. For the Animation Sync data type parameter modification is not possible, so all the new options are unavailable.

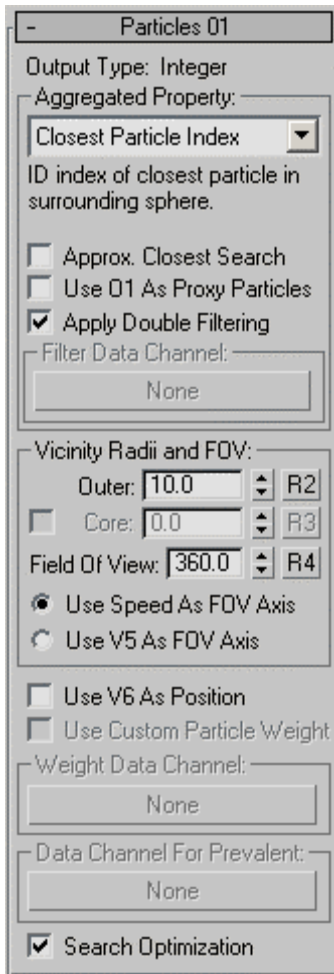
Particles Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Particles.

The Particles suboperator lets you derive a specific property for each particle, typically with respect to particles in its vicinity, such as the index of the particle closest to it within a spherical volume around the particle, or within a particular field of view. You can set the volume or field of view explicitly or input it from other suboperators.

The suboperator can work on the whole flow or particles from another particle flow, known as proxy particles.

Interface



The screenshot shows the 'Particles 01' suboperator interface. It features a title bar with a minus sign and the text 'Particles 01'. Below the title bar, the 'Output Type' is set to 'Integer'. The 'Aggregated Property' section contains a dropdown menu with 'Closest Particle Index' selected. Below this, a description reads 'ID index of closest particle in surrounding sphere.' There are three checkboxes: 'Approx. Closest Search' (unchecked), 'Use 01 As Proxy Particles' (unchecked), and 'Apply Double Filtering' (checked). The 'Filter Data Channel' is set to 'None'. The 'Vicinity Radii and FOV' section includes 'Outer' (10.0, R2), 'Core' (0.0, R3), and 'Field Of View' (360.0, R4). There are two radio buttons for 'Use Speed As FOV Axis' (selected) and 'Use V5 As FOV Axis'. Below these are two more checkboxes: 'Use V6 As Position' (unchecked) and 'Use Custom Particle Weight' (unchecked). The 'Weight Data Channel' is set to 'None'. The 'Data Channel For Prevalent' is also set to 'None'. At the bottom, the 'Search Optimization' checkbox is checked.

Output Type – This read-only field displays the data type of the active item in the Aggregated Property drop-down list. The data type for each property is shown in the description, following.

Aggregated Property group

Aggregated Property – Choose the property to be aggregated from particles. The property is aggregated from a

- **Closest Particle Distance** – The distance to the closest particle in the surrounding sphere. Data type=Real.
- **Closest Particle Index** – The ID number of the closest particle in the surrounding sphere. Data type=Integer.
- **Closest Particle Position** – The position of the closest particle in the surrounding sphere, in world coordinates. Data type=Vector.
- **Density** – The density of particles in the surrounding sphere measured in particles per cubic unit. Data type=Real.
- **Density Gradient** – The direction of the greatest change in particle density in the surrounding sphere. The length of the output vector indicates the slope, or rapidity, of the change in density. Data type=Vector.
- **Number of Neighbors** – The number of particles in the surrounding sphere. Data type=Integer or Real.
- **Number of Particles** – The number of particles in the current event or in proxy particle systems. Data type=Integer.
- **Prevalent Custom Data** – The average value of a custom data channel for particles in the surrounding sphere. Specify the custom channel with the [Data Channel For Prevalent](#) button. The data type can only be types that make sense for averaging: Real and Vector. If you have a different data type that you would like to use for querying prevalent data, you must convert it to Real or Vector first.
- **Prevalent Speed** – The average speed of the particles in the surrounding sphere. Data type= Vector.

Approx. Closest Search – When on, speeds up aggregation at the cost of precision. Available only with Closest Particle Distance, Closest Particle Index, and Closest Particle Position. Turn this on with very large quantities of particles. Default=Off.

Use O1 As Proxy Particles – When on, adds an Object-data input to the Particles suboperator. You can connect a particle flow from the Select Object suboperator to this input. The proxy particles are used for property aggregation. Default=on.

Use Filter For Proxies – Lets you input a Boolean data channel from a different operator to use as a filter. For example, you use this to restrict the particles to a single event. Use the Filter Data Channel button to specify the filter channel. Available only when Use O1 as Proxy Particles is on.

Apply Double Filtering – When off, aggregation is done for filtered particles with respect to all particles in the current flow. When on, aggregation is done for filtered particles with respect only to the filtered particles. Available only when Use O1 as Proxy Particles is off.

Filter Data Channel – Lets you specify a data channel for filtering proxy particles. Available only when Use Filter For Proxies is active.

Vicinity Radii and FOV group

Outer – The maximum distance from the particle within which particles are considered for aggregation. Unless you use Core as well (see following), weighting is always constant throughout this volume.

R2 – When on, lets you specify the Outer radius value by wiring a suboperator that outputs real-format data to the #2 input of the Particles suboperator.

Core – When on, greater priority is given to particles inside this radius. Inside the core radius, all particles are weighted 1.0 (100%). Outside the core radius, weighting falls off linearly to 0.0 at the Outer radius. For best results, this value should be less than Outer Radius value. Available only with the Density, Number of Neighbors, and Prevalent Custom Value/Speed properties.

R3 – When on, lets you specify the Core radius value by wiring a suboperator that outputs real-format data to the #3 input of the Particles suboperator.

Field of View – Lets you limit aggregation to particles within the volume of a cone whose tip, by default, is at the location of the particle being processed. By default, the value is 360.0, which specifies that all particles within a spherical volume are aggregated. To aggregate only particles within a hemisphere, set this to 180.0. For a quarter-sphere, use 90.0 (etc.). Use the FOV Axis parameters (see following) to set the direction of the field of vision.

R4 – Lets you specify the Field of View value by wiring a suboperator that outputs real-format data to the #4 input of the Particles suboperator.

Use Speed As FOV Axis – Sets the direction of the field of view to the direction in which particle is moving ("in front").

Use V5 As FOV Axis – Lets you specify the field of view direction by wiring a suboperator that outputs vector-format data to the #5 input of the Particles suboperator. For example, for looking behind, use an Input Standard suboperator set to Speed: Vector, and feed it through a Function suboperator with Type=Vector, Pre-Factor or Post-Factor=-1.0, and Use Second Operand=off).

Use V6 As Position – By default, the origin of the field of view is the position of the particle being processed. This option lets you specify a different position value by wiring a suboperator that outputs vector-format data to the #6 input of the Particles suboperator,

thus allowing you to find aggregated properties for a point other than the current particle position.

Use Custom Particle Weight – Use this if you need weighting values other than those defined by the Proximity/Core Radius settings.

For example, the Density option accounts only for the number of particles. Say you have different-size particles in the system and you want to calculate density taking particle mass into account, not just quantities. Create an additional channel that calculates the mass as the cube of particle size, and use that as weight data channel. Available for all properties other than Closest Particle Distance, Closest Particle Index, and Closest Particle Position.

Weight Data Channel – Lets you specify a data channel for defining custom particle weights. Available only when *Use Custom Particle Weight* is active.

Data Channel For Prevalent – Lets you specify a data channel for defining custom data for aggregation. Available only when Aggregated Property=Prevalent Custom Data.

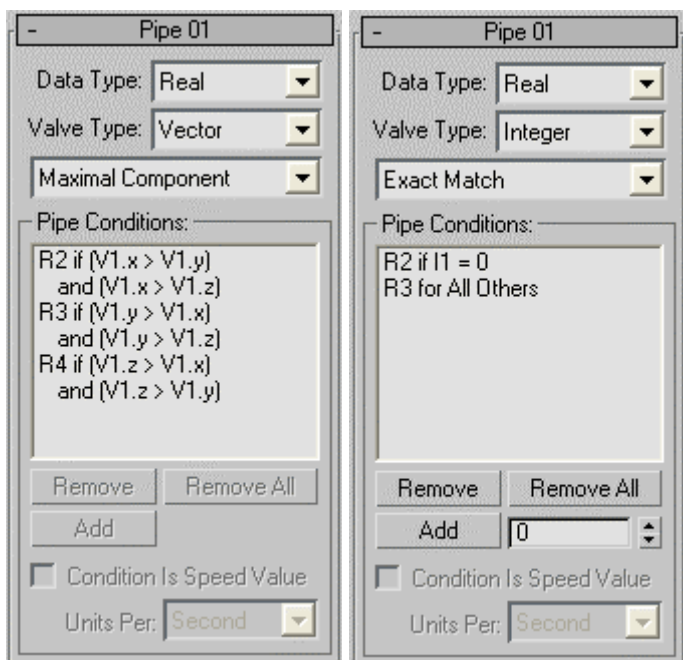
Search Optimization: Speeds up aggregation at the cost of memory. Turn on with large quantities of particles. Turn off to save memory with small quantities of particles. Default=on.

Pipe Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Pipe.

The Pipe suboperator lets you output one of several data inputs based on values in a "valve" stream. It provides at least two inputs: one valve (always the first input) and one or more data inputs. For example, the default Pipe suboperator has a Boolean valve and two Real data inputs, and outputs data from one or the other data input depending on the data coming into the Boolean input.

With other types of valves, you can set up ranges for the valve stream; the range in which the valve input value lies determines which data input is "piped" through. For example, you could have one Integer valve input (I1) and three Real data inputs (R2, R3, R4), and output R2 if $I1 \leq 0$; R3 if $I1 > 0$ and ≤ 50 ; or R4 if $I1 > 50$.



Interface

Data Type – Choose the data type for the incoming and outgoing "pipe" streams. Most of these are self-explanatory; for a discussion of the rest, see [Data Types](#).

Valve Type – Choose the data type for the incoming "valve" stream, which the suboperator uses to determine which data input is piped through. These are self-explanatory.

[drop-down list] – When Valve Type is set to Vector or Integer, this list becomes available and lets you choose the available pipe conditions.

In the case of Valve, the incoming values V1.x, V1.y, V1.z (components of the vector) become the valve controllers. The drop-down list lets you choose how the vector components are used for piping: Maximal Component, Maximal Absolute Component, Minimal Component, or Minimal Absolute Component. With the Vector valve type, the Add and Remove buttons become unavailable because the number of input data channels is fixed at three, similarly to the Boolean valve type, where the number of input data channels is fixed at two.

In the case of Integer, the drop-down list shows two choices: Intervals, which is the standard mode for most valve types, and Exact Match, which tests for equality of the specified values only.

Pipe Conditions

The list box in the Pipe Conditions group displays the conditions used to determine which data input gets piped through. For all valve types except Boolean, these are ranges based on values you specify. By default, if the valve type is not Boolean, a single condition (and input) exists, and is set to All, meaning that the input data is simply piped through. By adding conditions with the Add button, more

Remove – Deletes the highlighted condition from the list, and the corresponding input from the suboperator.

Remove All – Deletes all conditions in the list, restoring the original, single "All" condition.

Add – Adds a " \leq " condition to the list, based on the current value of the numeric field to the right of this button. For example, starting with a single "All" condition, if Data Type=Matrix, Valve Type=Float, and the numeric field is set to 0.5, after clicking Add, two conditions exist: M2 if $R1 \leq 0.5$, and M3 if $R1 > 0.5$.

The Pipe suboperator supports a maximum of seven conditions; after the seventh is added, the Add button becomes available. Also, conditions are automatically sorted in numeric, ascending order.

Condition is Speed Value/Spin Rate – This check box is available only when Valve Type= World Unit or Angle. When Valve Type=Angle and the check box is on (Condition is Spin Rate), the condition is considered to be angular velocity, rather than a specific angle. When Valve Type=World Unit and the check box is on (Condition is Speed Value), the condition is considered to be velocity, rather than a specific position.

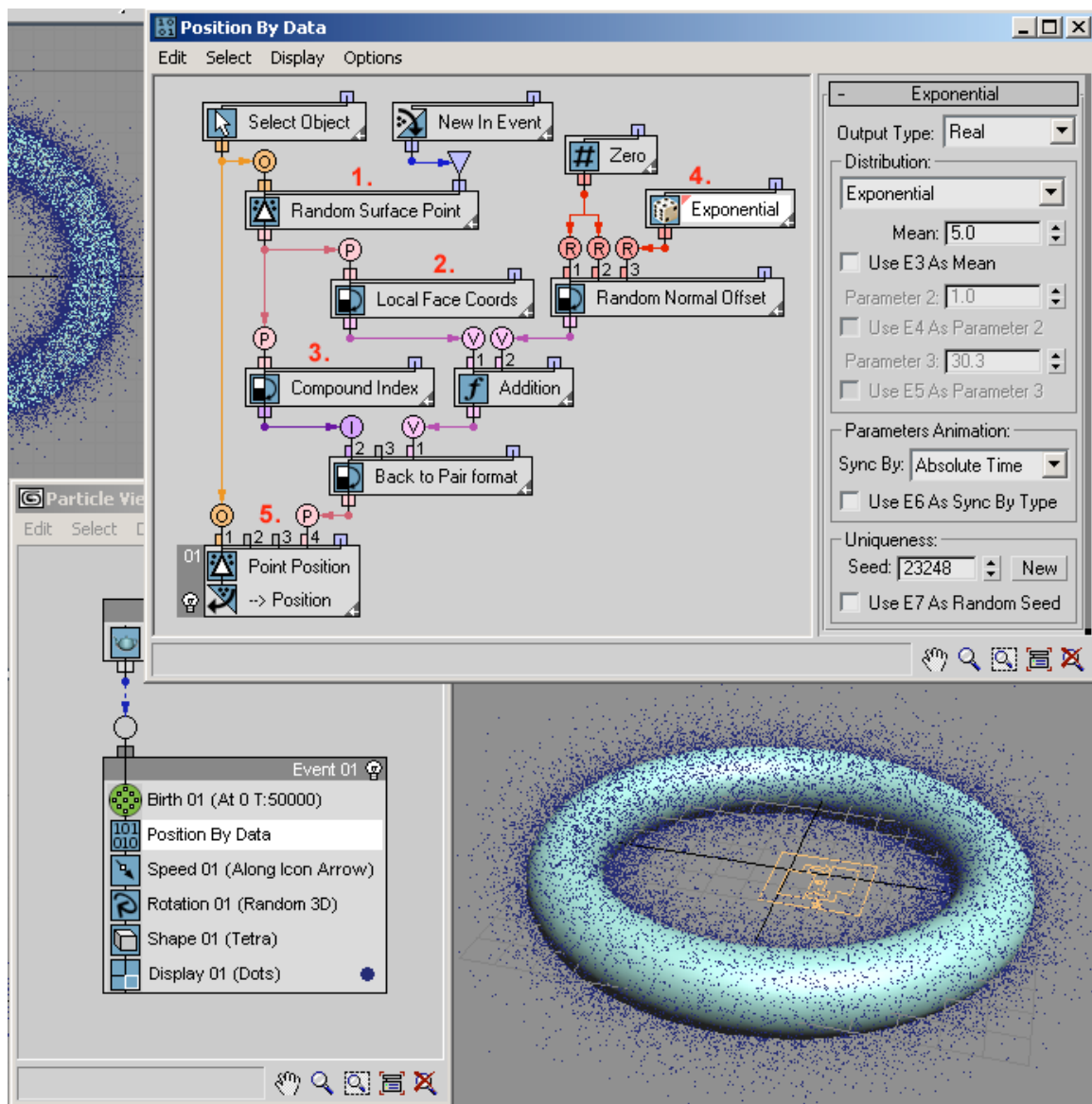
Units Per – Set to Frame, Second, or Tick.

Random Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Random.

Controlled chaos concept is the essence of a particle system. Therefore, to add some chaos into a Data operator, we use the Random suboperator. The Random suboperator generates random values in scalar and vector formats using a variety of algorithms, as described in this section.

Let's start with an example of how to place particles in chaotic manner surrounding a reference object: see included file *RandomPositioning.max*.



In the data flow illustrated above, the Geometry suboperator (1.) distributes random points uniformly over the surface of a reference object. Its Pair data output is then split off to two different Convert suboperators: Face Local Coordinates (2.) and Compound Index (3.) that includes the face index where a random point is located.

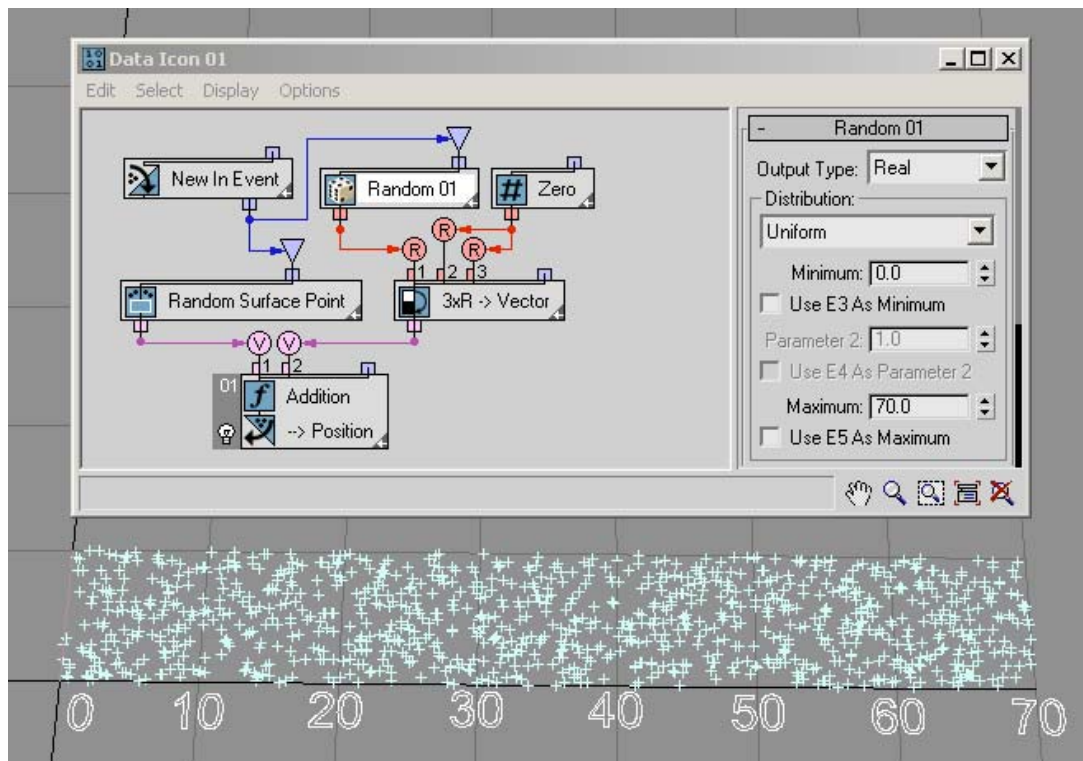
To obtain a desirable volume distribution of particles surrounding the reference object, we need to "lift off" the particles from the surface; this can be done by modifying the Z component of the face local coordinates vector. The Z component is the distance from the face in the direction of the face normal.

A Random suboperator (4.) can be used to define the amount of "lift off." For this example we used Exponential distribution, which is positive and denser around zero. Thus more particles are placed closer to the surface, and the density decreases progressively with the distance from the surface.

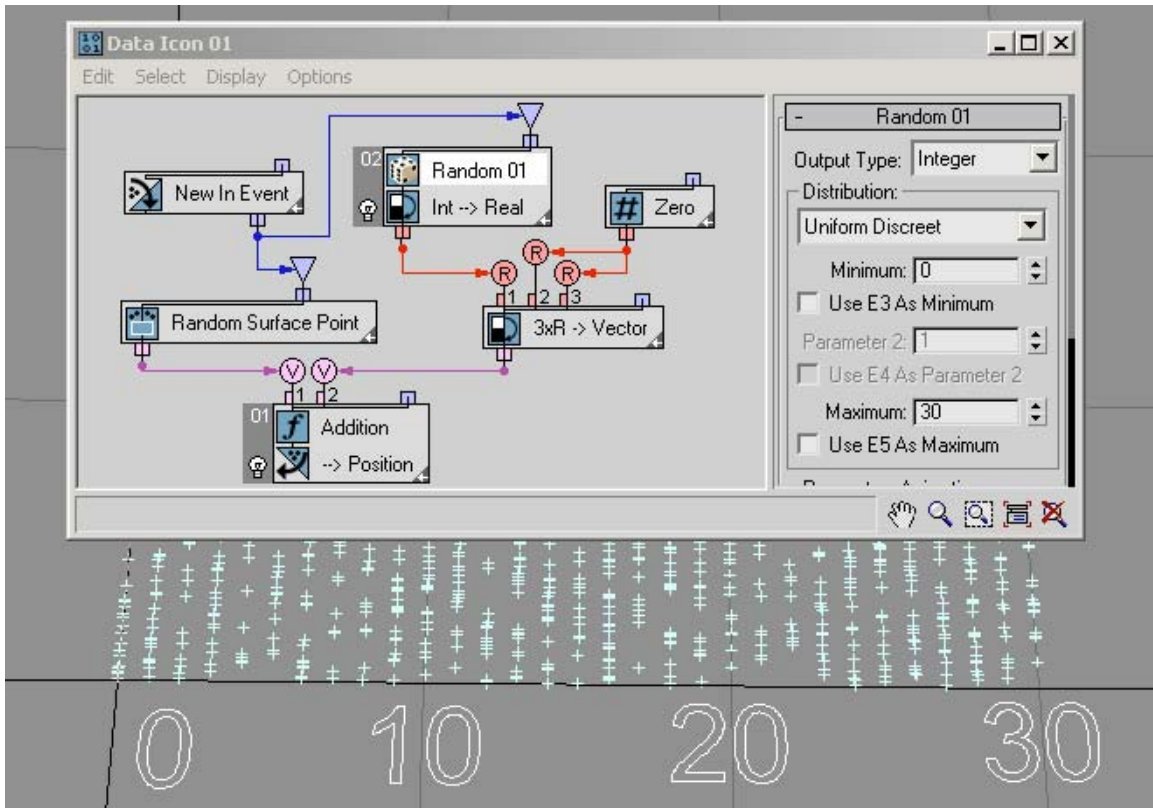
Then the modified face local coordinates value and the compound index are converted back into the Pair data format, which is used in the Geometry suboperator (5.) to calculate the particle position in world coordinates.

Let's look closely at the Distribution parameter of the Random suboperator. The parameter defines the type of the random function used in the suboperator. The function type also depends on the data type that Random suboperator produces.

For the next discussion, open the included scene file *RandomTemplate.max*, which we will use to play with different distribution functions.



Let's start with a simple distribution type: *Uniform Discrete*. This is the only type available if the Random suboperator is set to the Integer data type. *Discrete* is the opposite of *Continuous*. The Random suboperator generates integer data (in that sense it is discrete) that are uniformly distributed between the Minimum and Maximum values as defined in the UI. In the next example *RandomTemplate01.max*, the Minimum and Maximum values are set to 0 and 30. As a result, you can see that the particle positions are distributed along the lines of X ordinates: 0, 1, 2, 3, etc., up to 30. Here's more about the theory behind [Uniform Discrete distribution](#).

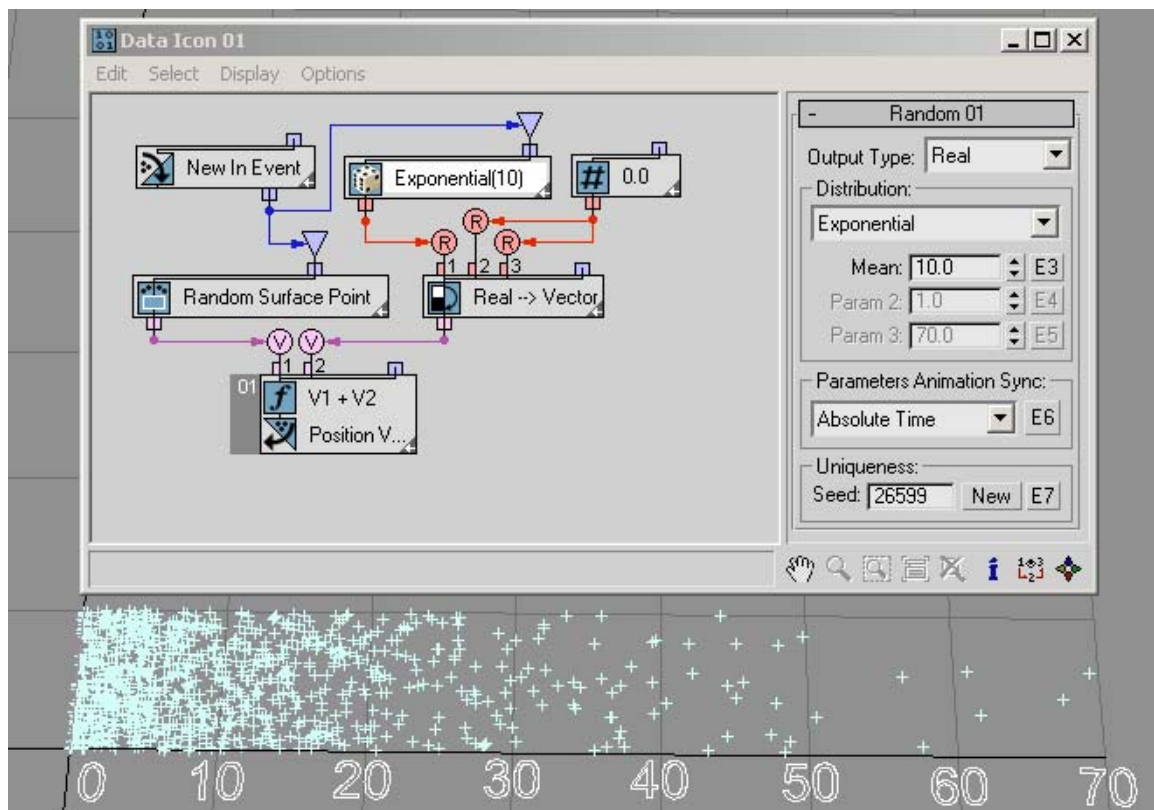


The Real data output type offers the largest variety of available distribution types.

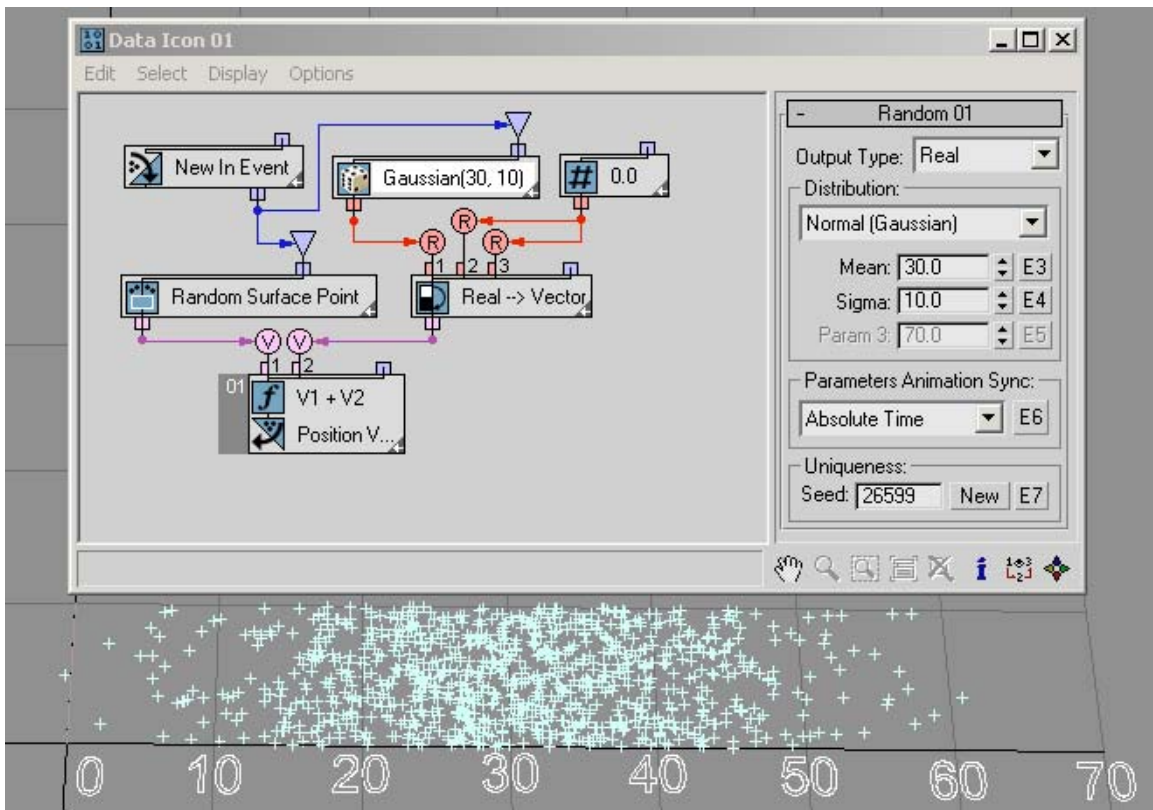
The *Uniform* distribution type is very similar to *Uniform Discrete* type – there are Minimum and Maximum values to define the range of distribution, but now the output values are continuously distributed in the range interval. Here's more about [Uniform distribution](#).

The *Exponential* distribution option is commonly used in reliability engineering. It can be used to model the behavior of units that have a truly random constant failure rate. Another example of exponential distribution is the distribution of individual lifetimes of unstable particles in radioactive decay.

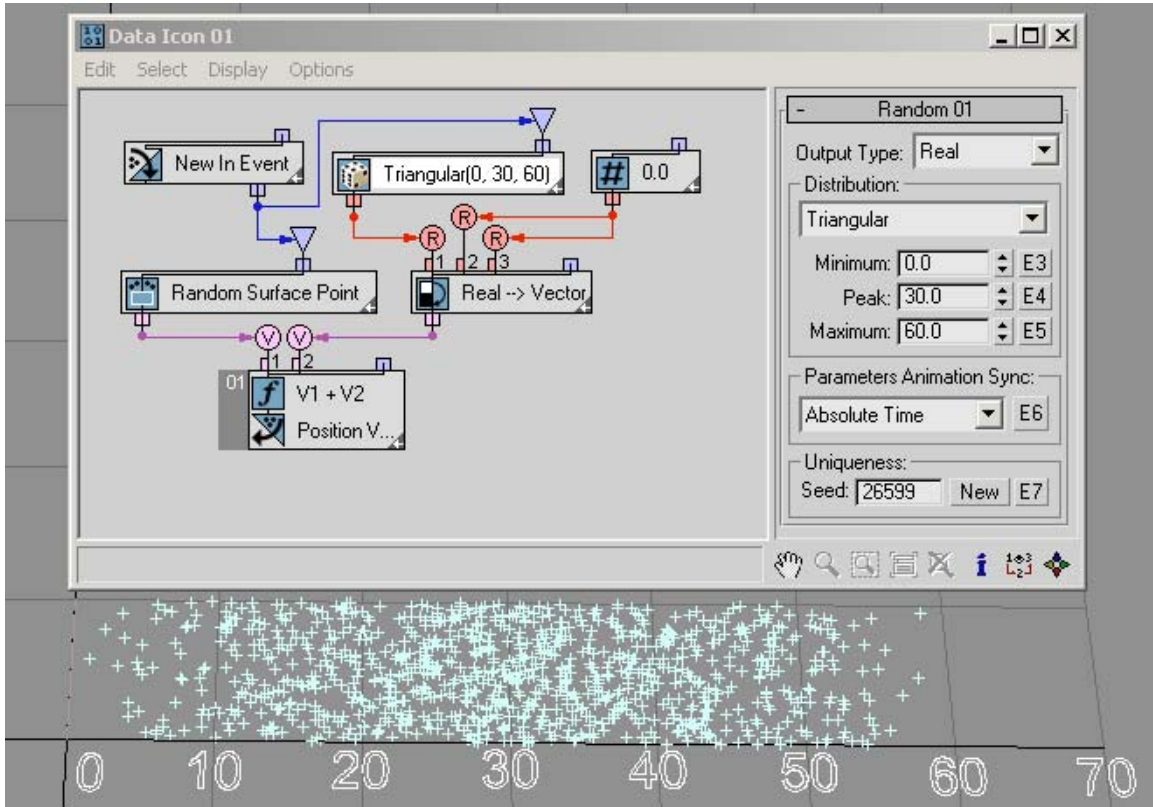
The Exponential distribution option generates positive values only. About half of the generated values are less than the Mean parameter value; with more values generated closer to zero. A generated value can be arbitrary large; however, the larger the value the less likely it is to be generated. In this example *RandomTemplate02.max*, the Mean value is 10; as a result, most of the generated values are between 0 and 70. Here's more about [Exponential distribution](#).



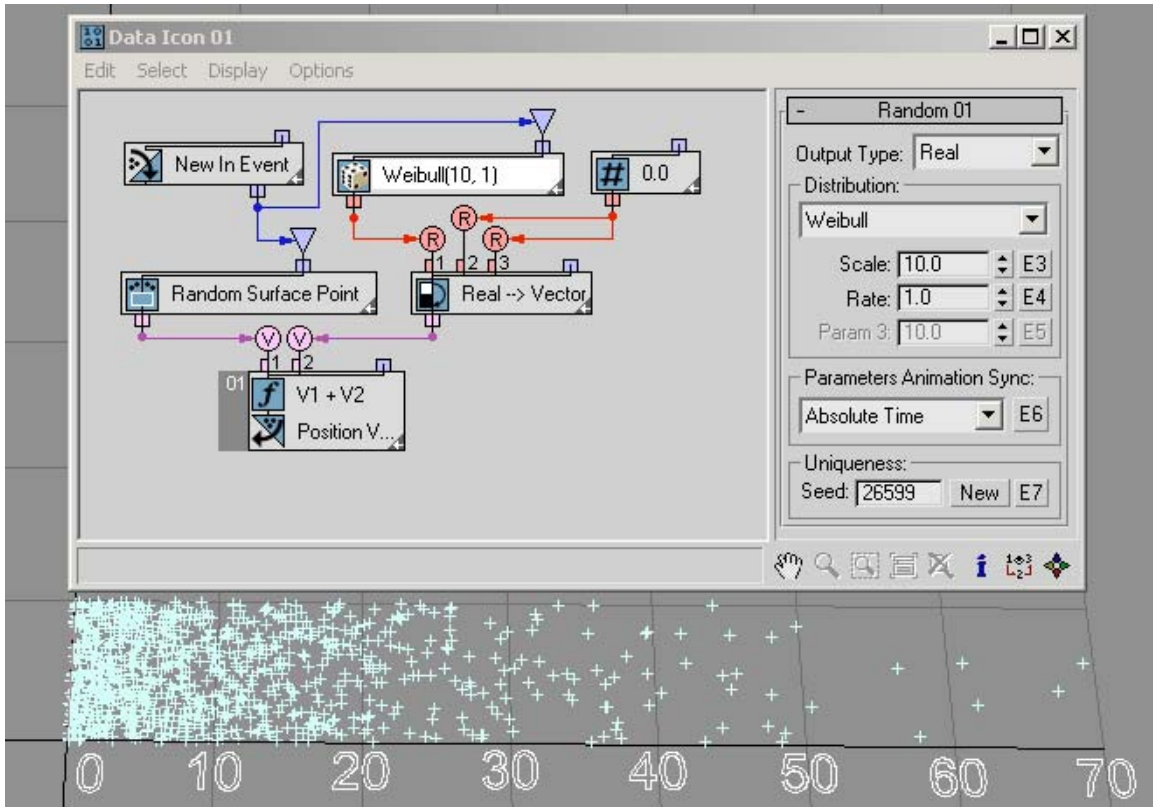
The *Normal*, or *Gaussian*, distribution option describes the distribution of independent random errors of observation. Other things that typically take on a normal or close-to-normal distribution include body temperature, shoe sizes, diameters of trees, etc. A normal-distribution graph is a symmetrical, bell-shaped curve. The Mean parameter defines the average value of the generated values (the center of the bell curve), and the Sigma parameter defines how far the generated values can fall out of the average value. The majority of the values (99.7%) are generated inside the 3 Sigma offset interval $[\text{Mean} - 3 * \text{Sigma}, \text{Mean} + 3 * \text{Sigma}]$. In the following example *RandomTemplate03.max* the Mean parameter is 30 with the Sigma parameter set to 10, hence the majority of values generated fall into the interval $[0.0, 60]$. Normal distribution generates negative and positive values. A generated value can be arbitrary large; however the larger the value the less likely it is to be generated (see the 3 Sigma rule, above). Here's more about [Normal distribution](#).



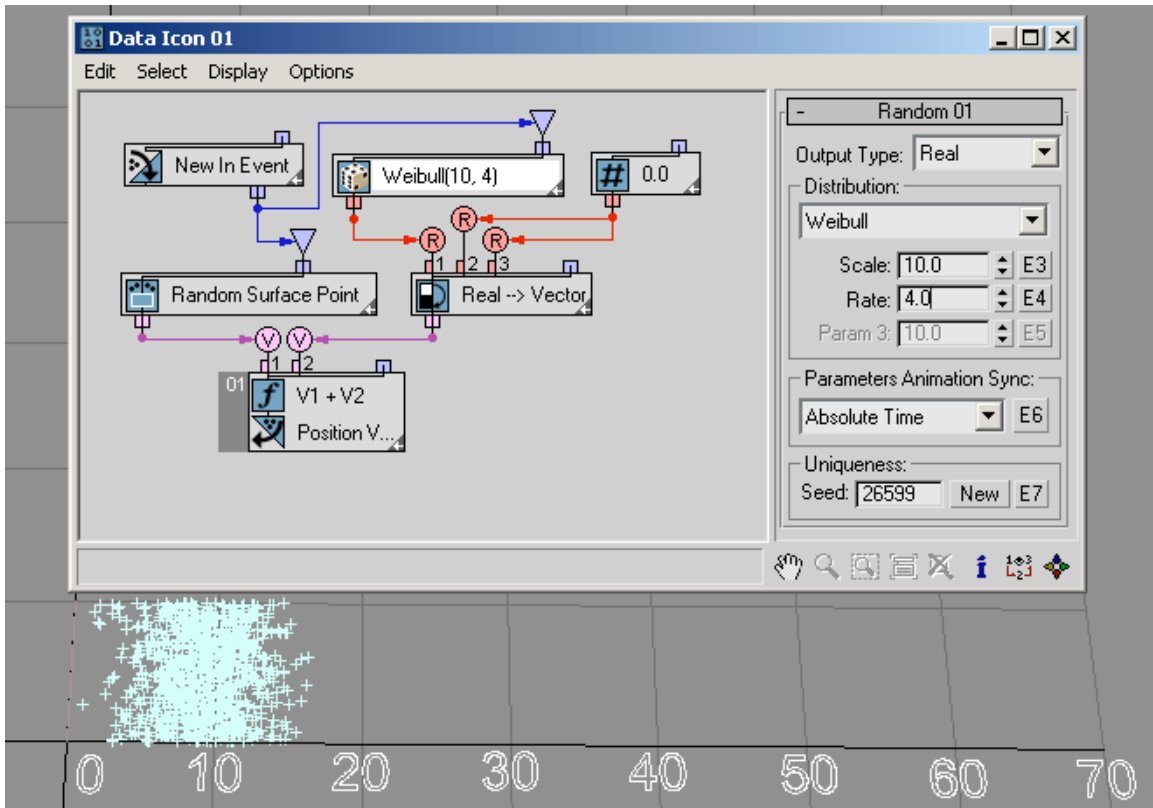
The *Triangular* distribution option is a simplistic way to describe a random phenomenon that tends to some Peak value and is limited in range by Minimum and Maximum values. In this example *RandomTemplate04.max*, Triangular distribution emulates the Normal distribution from the previous example. Here's more about [Triangular distribution](#).



The *Weibull* distribution method extends Exponential distribution to events that are not purely random; the distribution is commonly used in reliability and lifetime-with-aging modeling. In this example *RandomTemplate05.max* the Rate parameter is equal to 1.0, thus making the Weibull distribution equal to the Exponential distribution with the Mean parameter as the Weibull Scale parameter.



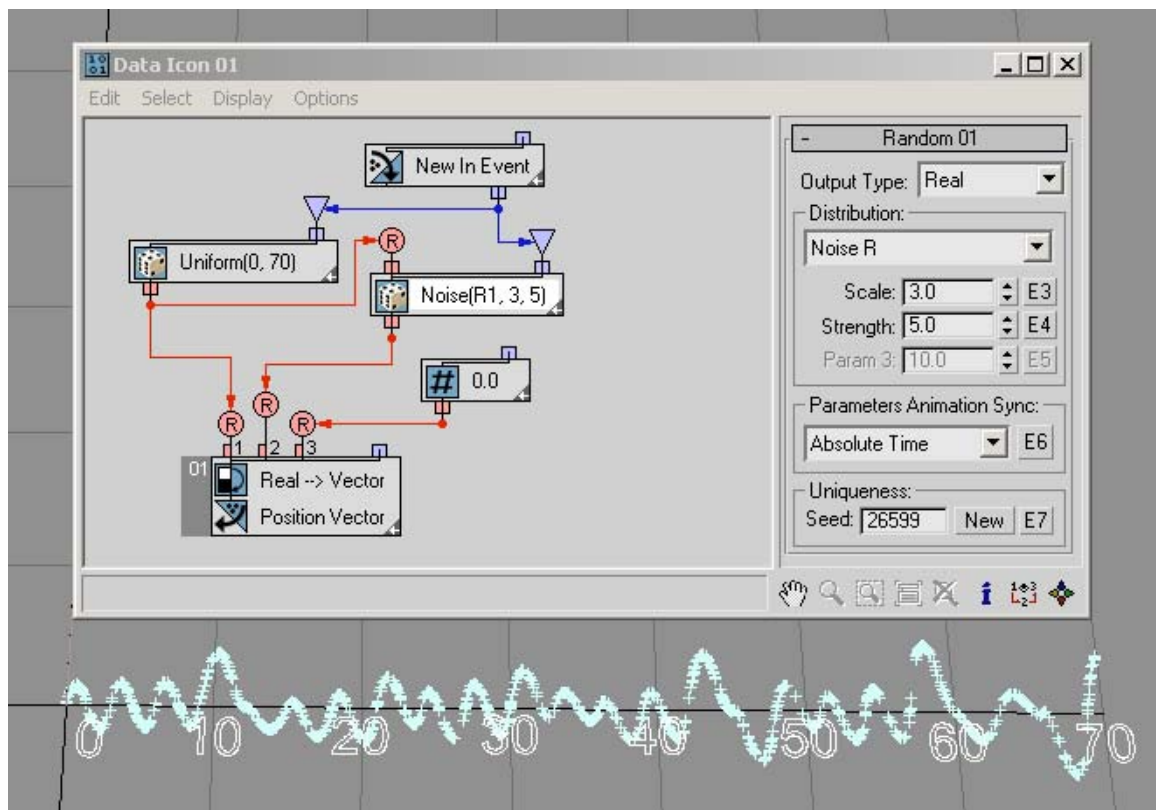
You can use the Weibull distribution to model the time until a given technical device fails. If the failure rate of the device decreases over time, set Rate < 1. If the failure rate of the device increases over time, set Rate > 1. Weibull distribution can also be used to model the distribution of wind speeds at a given location on Earth -- every location is characterized by a particular Rate and Scale parameter. In this example *RandomTemplate06.max*, the failure rate is equal to 4 and Scale is equal to 10, making the majority of the random values to be generated in the interval [3.0, 15.0]. Here's more about [Weibull distribution](#).



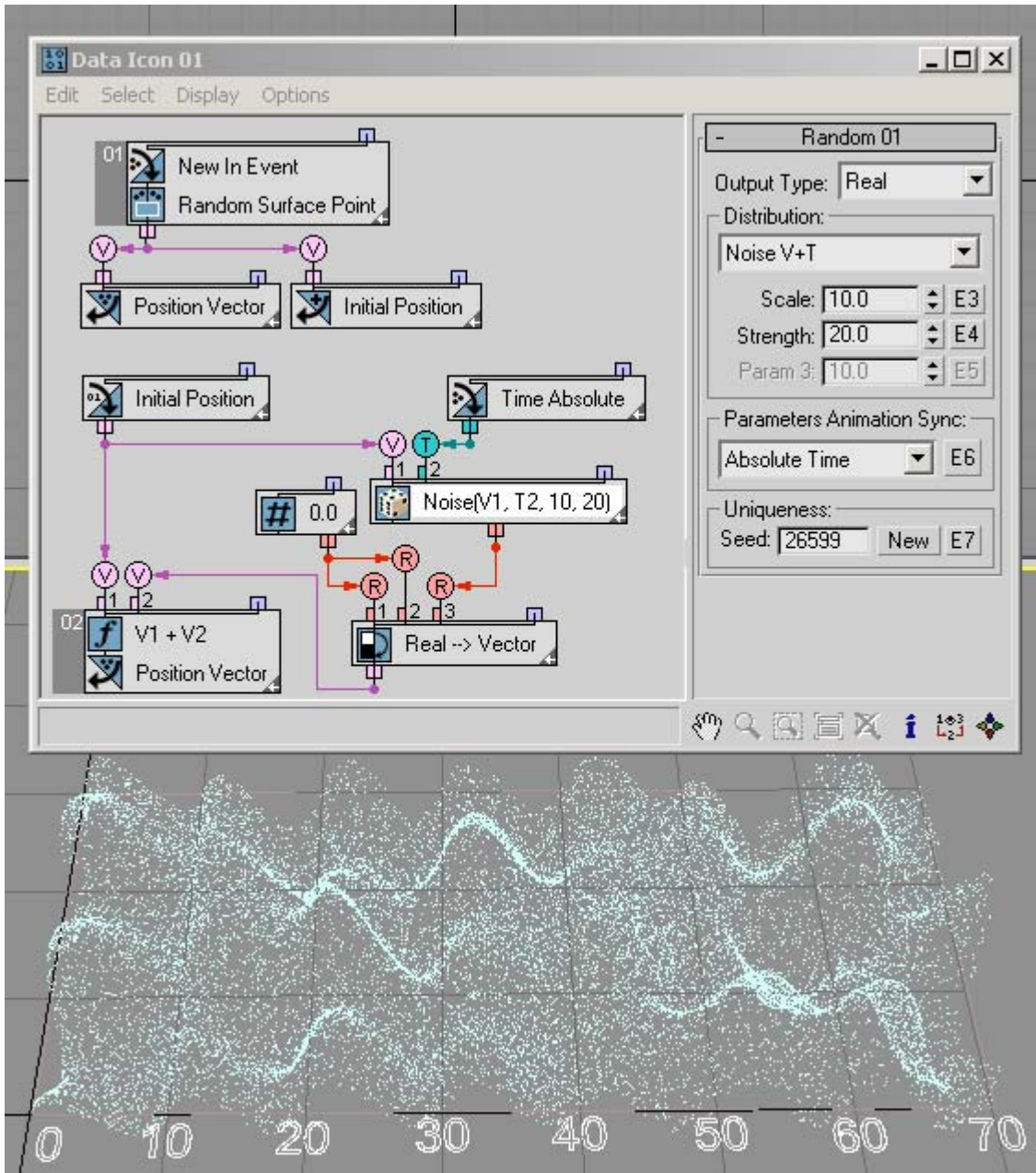
You can use the *Noise R*, *Noise V*, *Noise V+T*, *Turbulence V*, and *Turbulence V+T* options to generate pseudo-random noise-type values on the basis of input real, vector and/or time data. The Scale parameter sets the dependency rate between input and output data. Larger values produce smoother noise, lower values more jagged noise. The Strength parameter controls the magnitude of the output values.

The Turbulence options have an Iterations parameter that controls the number of iterations, or octaves, used to generate the fractal noise. Lower Iterations values create smoother output. The Iterations parameter has range from 1.0 to 10.0. The Noise options generate positive and negative output values; the Turbulence options generate positive values only.

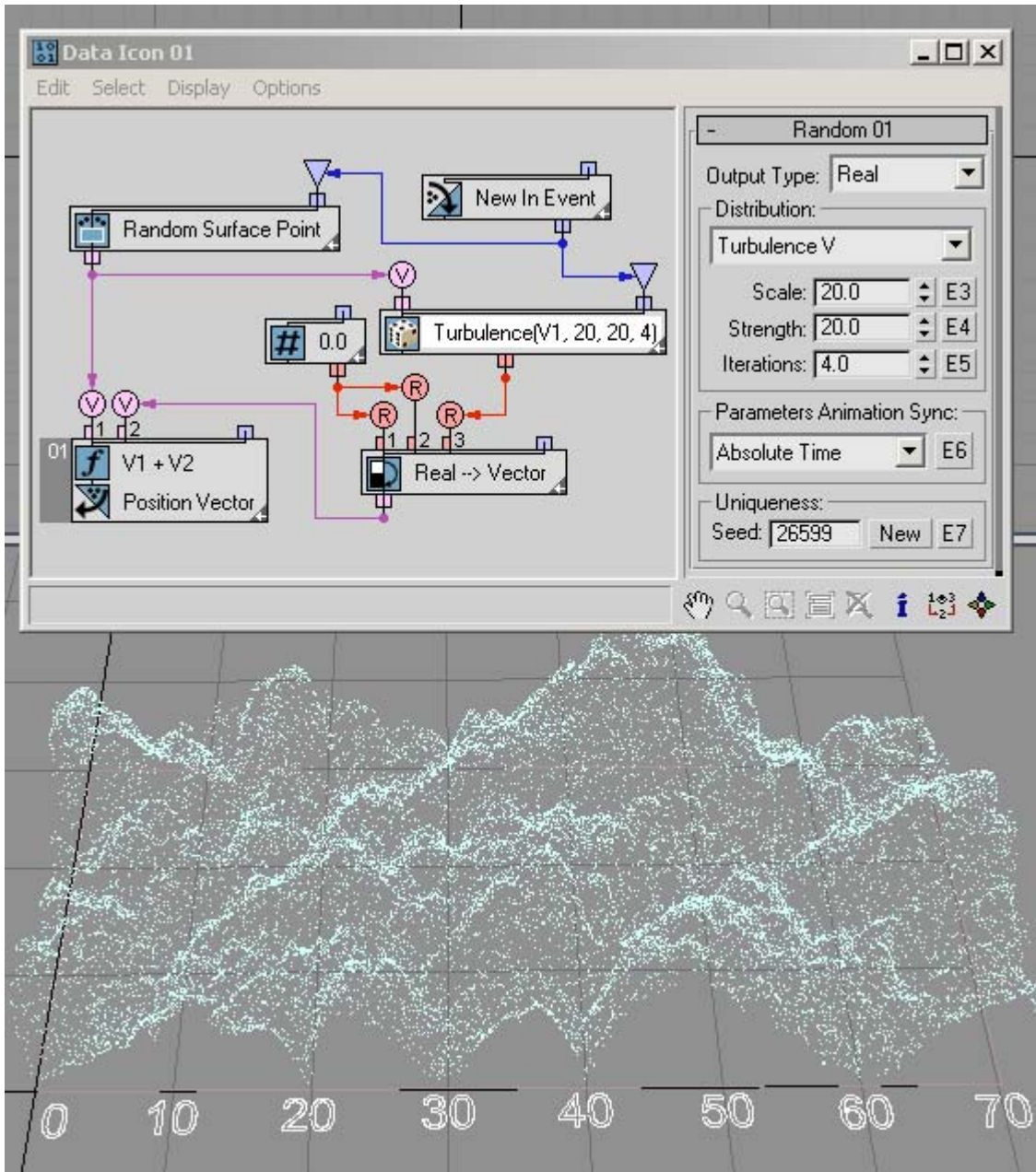
This example *RandomTemplate07.max* uses a random X coordinate of a particle to generate noise-like random values for the Y coordinate:



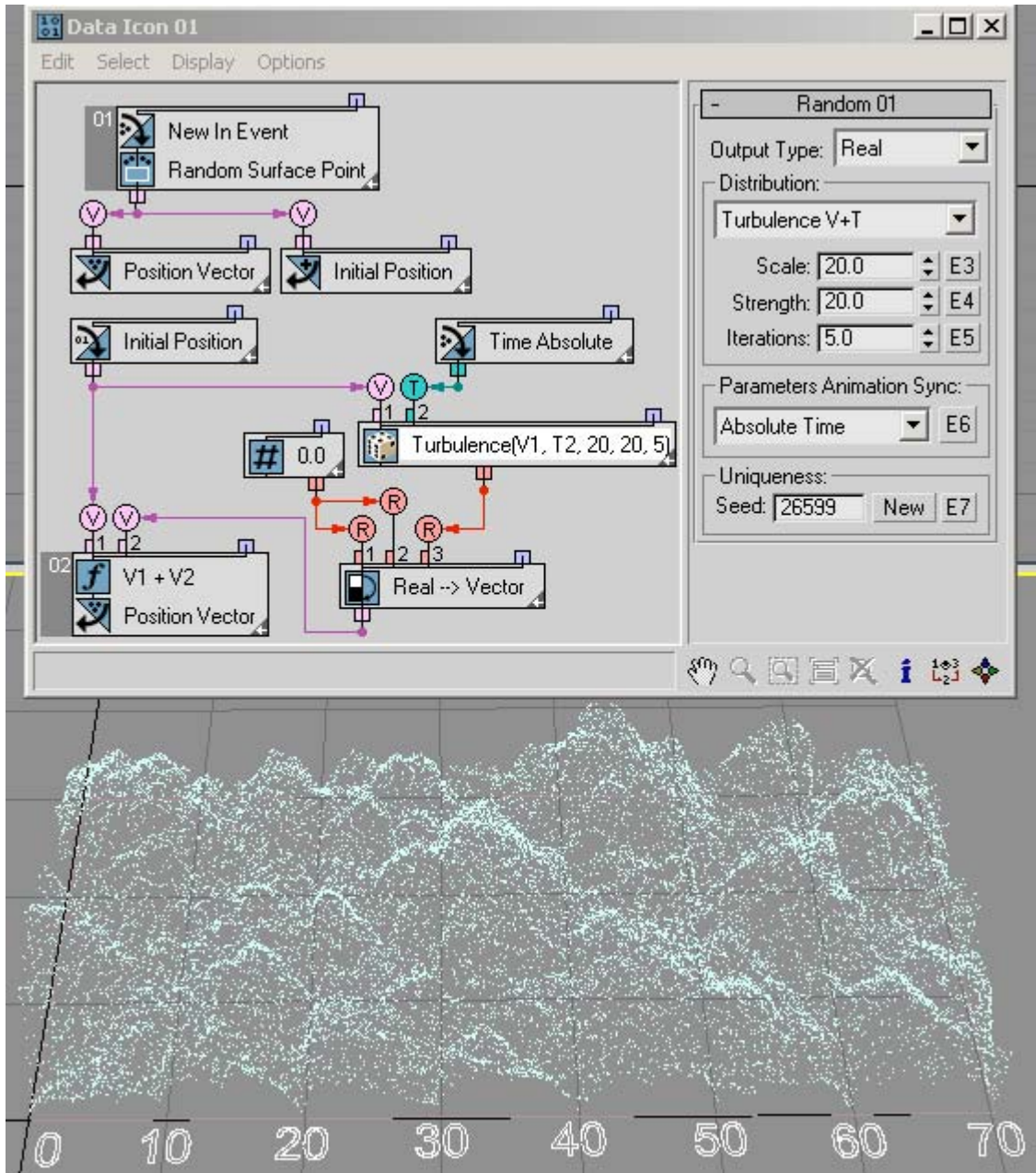
By wiring Absolute Time into the Time input of the ...+T options the generated noise can be easily animated as shown in this example *RandomTemplate09.max*:



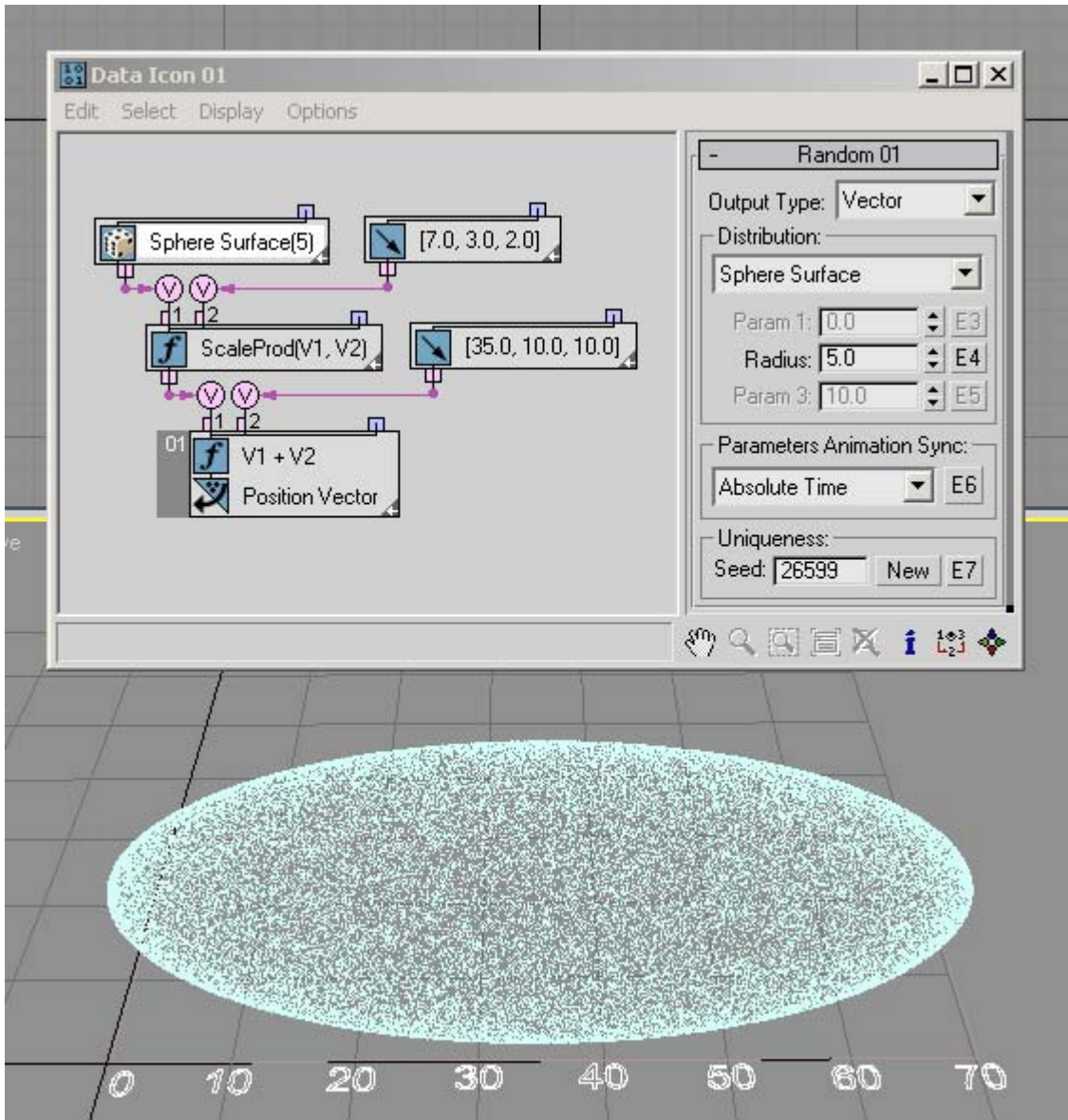
The Turbulence options are quite similar to the Noise options. You just need to define the Iterations parameter, as shown in this example *RandomTemplate10.max* ...



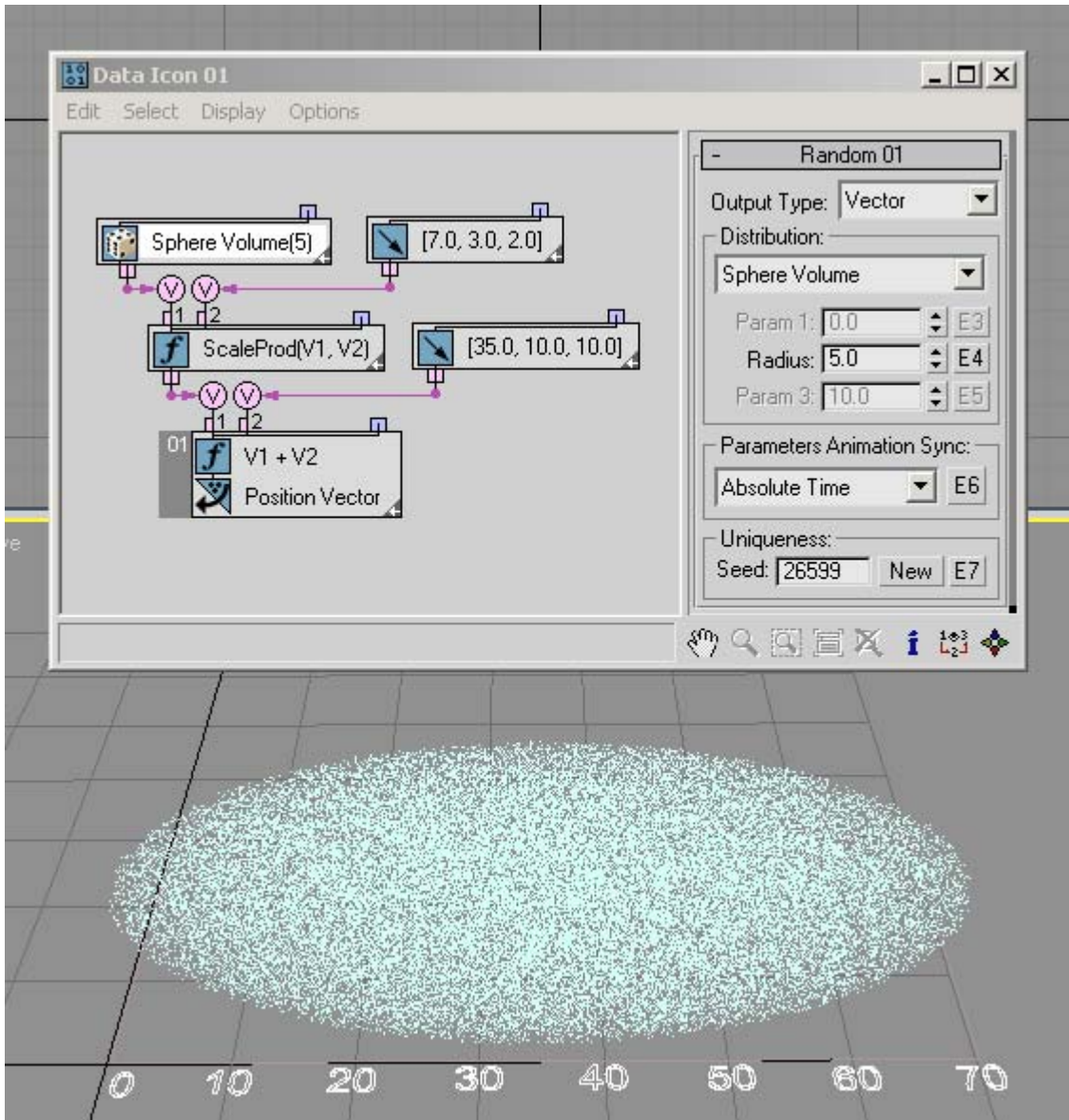
... and this example *RandomTemplate11.max*:



The Sphere Surface option can be used to place particles on the surface of a sphere. However, the most common usage is the generation of a vector with random direction. In this case, the Radius parameter defines the length of the vector *RandomTemplate13.max*:

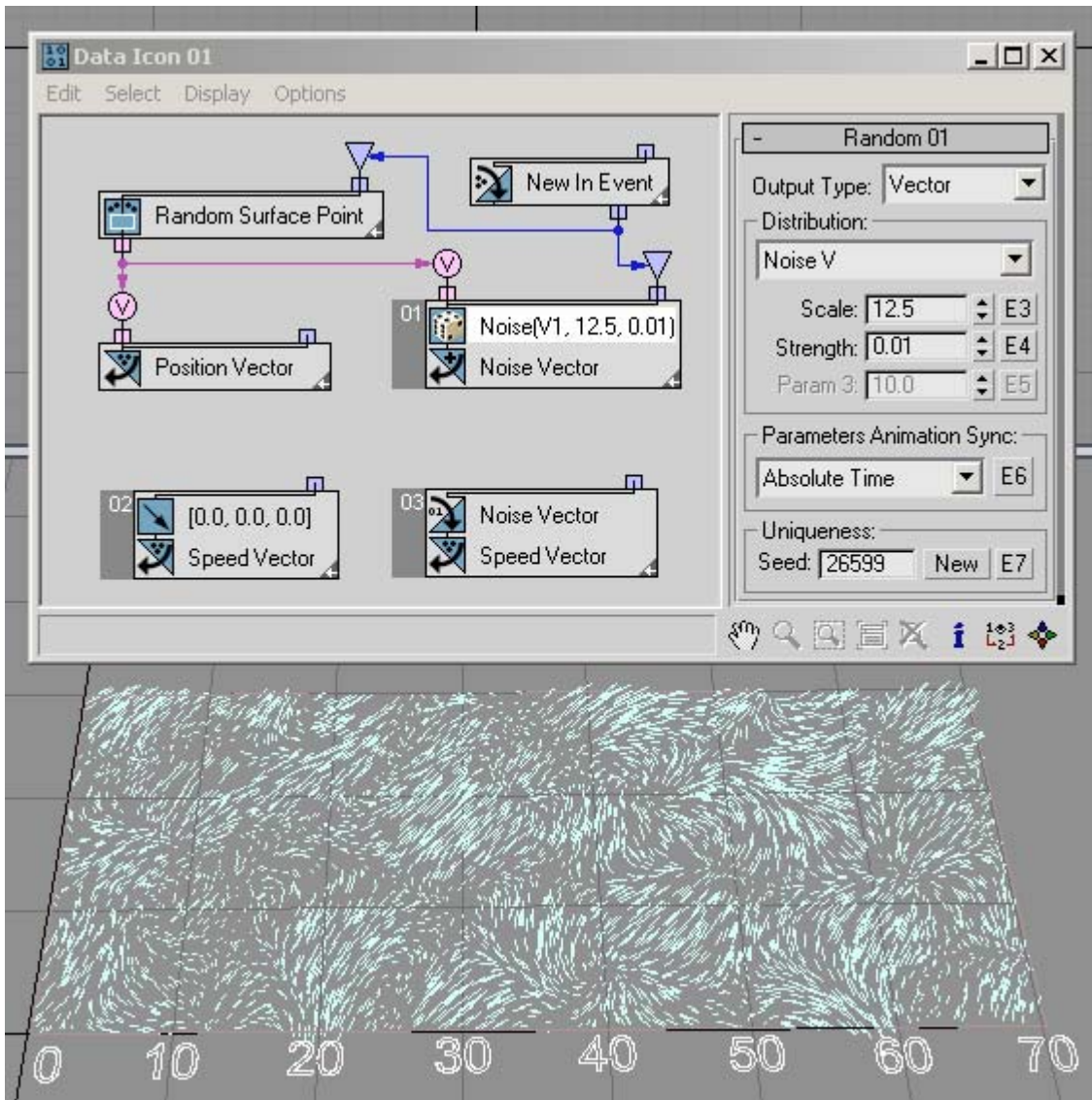


Or you might want to fill the whole Sphere Volume with random points
RandomTemplate14.max:

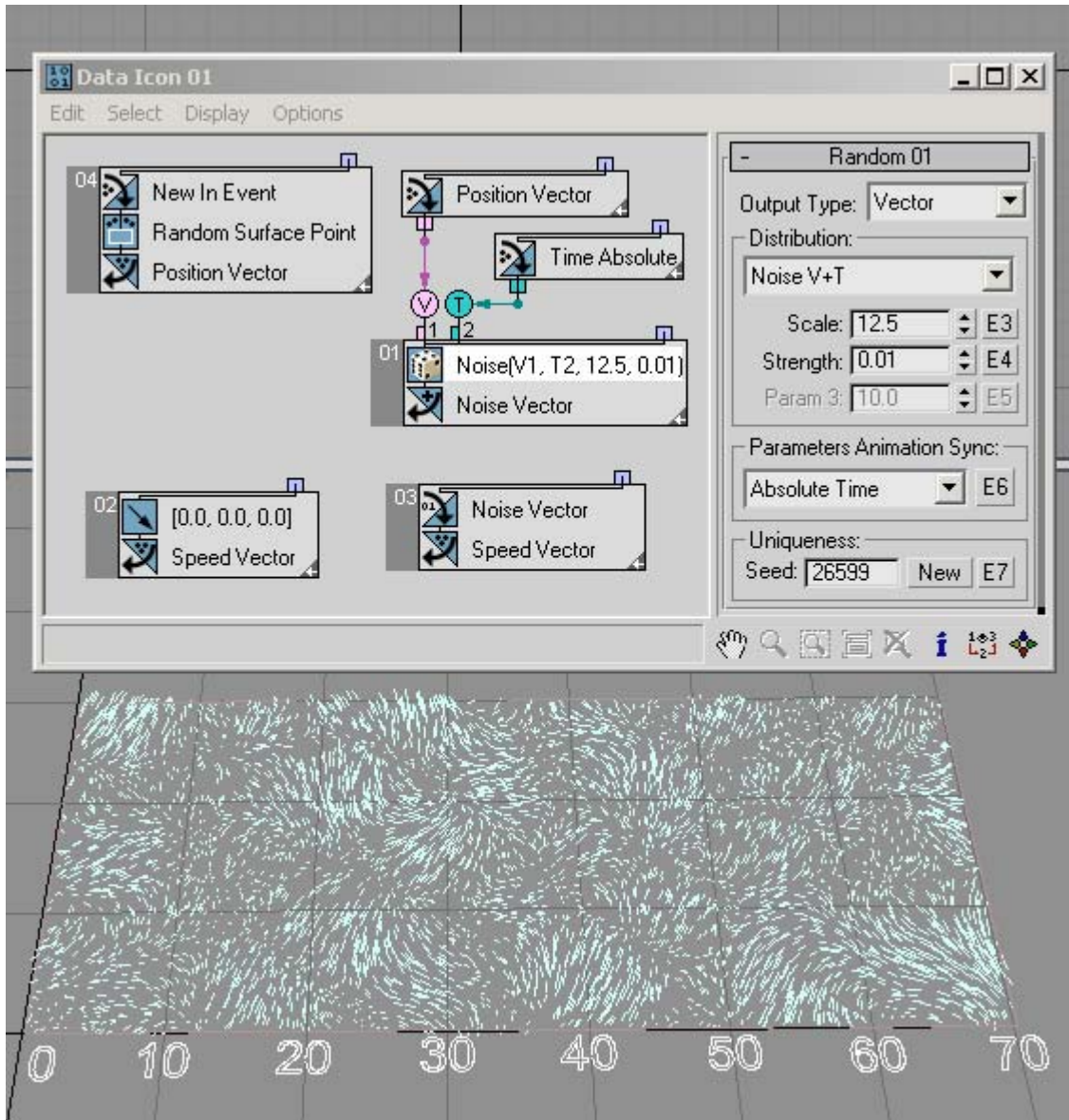


You can use the Normal (Gaussian) option to fill a sphere volume as well, but with this option there is no boundary for the sphere--more points are generated closer to the sphere center, and they diminish outwards. The same 3 Sigma rule as before can be applied here: 99.7% of the points are generated inside a sphere with radius equal to 3*Sigma
RandomTemplate15.max.

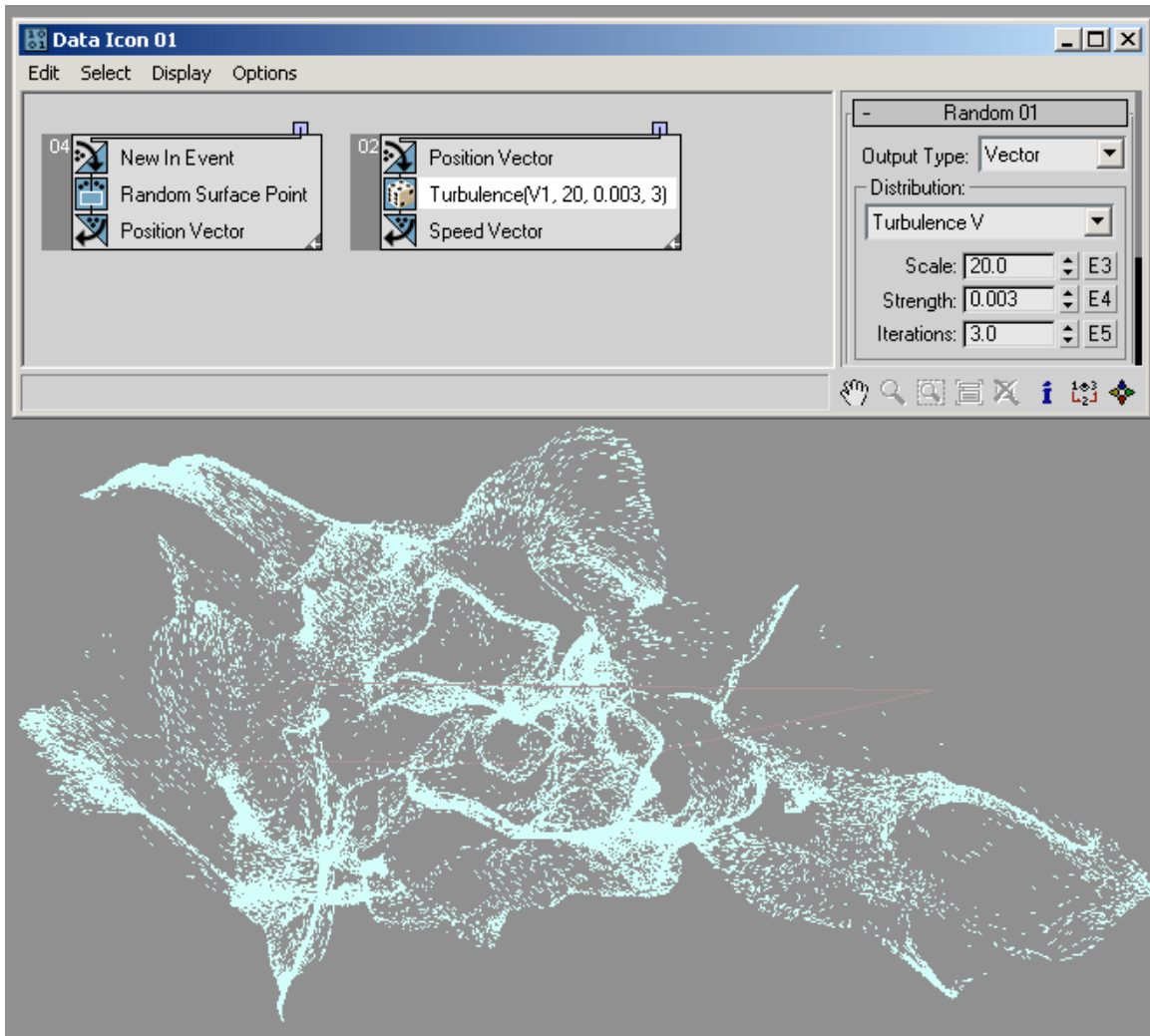
The *Noise V*, *Noise V+T*, *Turbulence V*, and *Turbulence V+T* options are similar to the Real-data options of the same names, except that they generate Vector data. To illustrate the output data, we can plug it into the Speed channel in Post step, and set speed to zero at Pre step. This way the particles don't go anywhere but we can draw the output as speed lines. In this example *RandomTemplate16.max* the particles are randomly placed in a rectangle, and then their position is used as an input for the Random suboperator with the Noise V option:



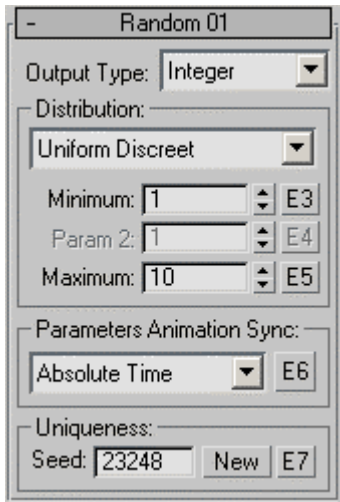
And here *RandomTemplate17.max* is the same setup, but with the Noise V+T option. We use current time as input for the Random suboperator to have animation for the Noise output:



We can wire the Random output into the Speed channel, and use the Position channel as an input for the Random suboperator – it can be used to simulate wind turbulence. This is frame 98 of the scene file *RandomTemplate18.max*:



Interface



Output Type – Choose the data type to output:

- Integer
- Real
- Vector

The Output Type setting determines the parameters available in the Distribution group.

Distribution group

The contents of the Distribution drop-down list are determined by the Output Type choice. The choice of Distribution in turn determines the remaining parameters in the Distribution group.

See the discussion, above, for specific information about the Distribution options and related parameters.

Parameters Animation Sync – If you animate the suboperator parameters, the software can begin applying this animation to all particles as of the start frame of the animation or the first frame of the current event, or to each particle based on its age. The options are:

- **Absolute Time** – Any keys set for parameters are applied at the actual frames for which they're set.
- **Event Duration** – Any keys set for parameters are applied to each particle relative to the frame at which it first enters the event.
- **Particle Age** – Any keys set for parameters are applied at the corresponding frames of each particle's existence.
- **Particle Lifespan** – Scales/maps the animation of the parameters onto the particle lifespan period. For example, if a parameter value is animated from 5-20 over

frames 0-100, then this parameter has the value 5 when the particle is born, and the value 20 when the particle dies. This way you can, for example, define the change in a particle's scale over its lifespan.

For this option to work properly, there must be a Delete operator set to By Particle Age in the flow to define particle lifespan.

- **Input Proxy** – Adds a Time input to the suboperator, to which you can link any other suboperator that outputs data in Time format.

E6 – Adds an [Equal-type](#) data input for controlling the Animation Sync value. This can receive input only from a [Parameter suboperator](#) set to Type=Animation Sync.

Uniqueness – Lets you vary the random-number sequence generated by the Random suboperator. Enter a Seed value manually or click New to let the software generate a Seed value.

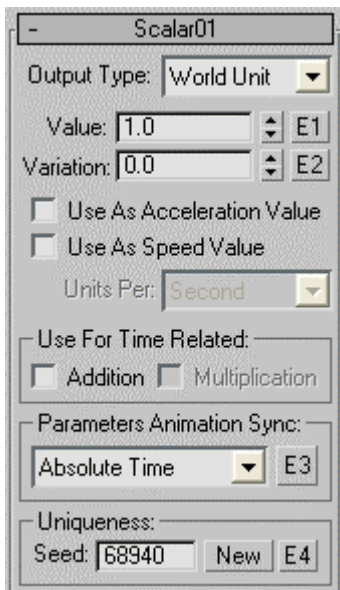
E7 – Adds an [Equal-type](#) data input for controlling the Uniqueness value. This can receive input only from a [Parameter suboperator](#) set to Type=Uniqueness.

Scalar Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Scalar.

The Scalar suboperator generates scalar data; that is, numerical values that indicate only magnitude, not direction. To generate vector data (magnitude + direction), use the [Vector suboperator](#).

Interface



Output Type – Sets the Data type of the output scalar value. Choose the type from the drop-down list; the choices are self-explanatory.

Value – Set the output value explicitly, or turn on E1 and wire a Parameter suboperator to the Scalar suboperator to let the user specify the value. Available with all output types except Boolean.

Variation – Set a value within which the output value will vary explicitly, or turn on E2 and wire a Parameter suboperator to the Scalar suboperator to let the user specify the variation. Available with all output types except Boolean.

True/False – Set the Boolean output value. Available only with the Boolean output type.

Use As Acceleration Value/Speed Value – When Output Type=World Unit, you can choose either of these to cause Particle Flow to use the scalar data as an acceleration or speed rate, in units per frame, second, or tick. You can activate only Acceleration Value or Speed Value, not both; clicking again turns the option off.

Use As Spin Rate – When Output Type=Angle and this check box is on, Particle Flow uses the scalar data as a spin rate, in units per frame, second, or tick.

Units Per – Sets the time frame for the acceleration, speed, or spin rate data.

Use For Time Related – These options let the suboperator take the integration step size into account in determining the output value. With the Float, Percent, or Time output type, you can choose Addition or Multiplication. Choosing one turns the other off. To turn both off, click the active check box. With the Angle or World Unit output type, you can choose either Use As Spin Rate/Speed Value (respectively) or Addition. Choosing Addition sets the Units Per setting to Time Unit; the options are the same.

Normally, the suboperator generates the value indicated by the Value setting. However, when Addition is on, the value is adjusted to the current integration step size. For example, if Time Unit is set to Frame, and the integration step size is a single frame, then the suboperator outputs the Value value. If the integration step size is a quarter frame, then the output value is one fourth of the Value value. This way, when the value is used for addition operations, the result is accumulated over several integration steps, and equals the Value setting.

If the Scalar suboperator is to be used for multiplication (such as slowing a particle by multiplying its speed by a coefficient less than 1.0, then use the Multiplication option. This also takes into account the integration step size to adjust the suboperator output. For example, say you want to slow a particle by a coefficient of 0.25 in a single frame, so you set Value=0.25. If the integration step size is a single frame, then the suboperator output is 0.25. If the integration step size is a half frame, then the output is 0.5. This value, when used twice for multiplication (0.5 x 0.5) equals 0.25, which is the explicit output setting for Value.

Parameters Animation Sync – If you animate the suboperator parameters, the software can begin applying this animation to all particles as of the start frame of the animation or the first frame of the current event, or to each particle based on its age. The options are:

- **Absolute Time** – Any keys set for parameters are applied at the actual frames for which they're set.
- **Event Duration** – Any keys set for parameters are applied to each particle relative to the frame at which it first enters the event.
- **Particle Age** – Any keys set for parameters are applied at the corresponding frames of each particle's existence.
- **Particle Lifespan** – Scales/maps the animation of the parameters onto the particle lifespan period. For example, if a parameter value is animated from 5-20 over frames 0-100, then this parameter has the value 5 when the particle is born, and the value 20 when the particle dies. This way you can, for example, define the change in a particle's scale over its lifespan.

For this option to work properly, there must be a Delete operator set to By Particle Age in the flow to define particle lifespan.

- **Input Proxy** – Adds a Time input to the suboperator, to which you can link any other suboperator that outputs data in Time format.

E3 – Adds an [Equal-type](#) data input to let the user control the Animation Sync value. This can receive input only from a [Parameter suboperator](#) set to Type=Animation Sync. Use [Expose Parameters](#) to make the setting available in the Particle View interface.

Uniqueness group

Seed – Lets you set a random seed to differentiate behavior from other randomly seeded functions. Change the Seed value to vary the random variation, either by entering a value or clicking New. Available with all output types except Boolean.

E4 – When on, you can expose the Seed parameter via a [Parameter suboperator](#) and let the user choose it. Turn on E4, add a Parameter suboperator set to Type: Uniqueness Seed, wire it to the E4 input on the Scalar suboperator, and then use [Expose Parameters](#) to make the setting available in the Particle View interface.

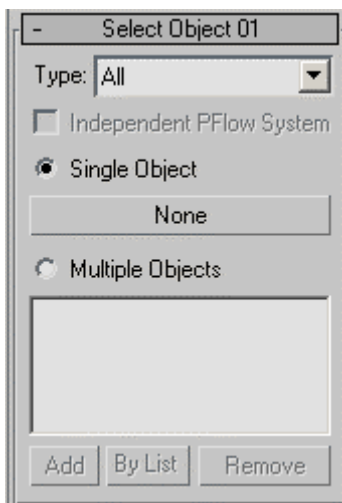
Select Object Suboperator

Path: Particle View > Data Icon/Data Icon Test > Click Edit Data Flow > Add or select Select Object.

If your operator/test requires one or more objects in the 3ds Max scene as parameters, use Select Object to specify the objects. Select Object outputs the Object data type, and is typically wired to the input of an Object or Geometry suboperator.

Note: Some suboperators such as Geometry can work on an Object Index property; that index is determined by the order of the object in the list in this suboperator. The first object in the list has index 0; the second has index 1, and so on.

Interface



Type – Lets you specify the type of objects that can be chosen. The object choice is an exposable parameter, so this lets you prevent the user from specifying an undesirable object type. The choices are:

- **All:** Any object in the 3ds Max scene can be chosen, including non-geometry objects such as lights and space warps, but excluding particle systems.
- **Geometry:** Only geometry objects can be chosen.
- **PFlow Systems:** Only Particle Flow systems can be chosen.

Independent PFlow Systems – The Select Object suboperator allows using other particle flows as reference objects for data gathering. This can create problems if two particle flows reference each other because it constitutes a bad reference loop. To resolve this situation, use this option.

When on, before gathering the data, the reference particle flow is updated to the current frame, and only that the data is taken. When off, there is no attempt to update the reference particle system before data gathering.

If you have two particle systems that depend on each other, first use the Cache Disk operator to pre-calculate the overall particle system states for all frames. Otherwise the history-dependent particle flows with reference loops might be inconsistent for network rendering.

Single Object – When chosen, you can specify a single default object in the scene by clicking the button, labeled "None" by default, and then selecting an object in the viewport. If you expose this parameter, the button is available to the user.

Note: Selecting an object in a group specifies the whole group. To specify group members, choose Multiple Objects, click the Add button, and then select the object.

Multiple Objects – When chosen, you can add any number of objects, either by clicking Add and then selecting an object to add to the list, or clicking By List and highlighting as many objects as you like. Use Remove to delete highlighted items from the list.

If you use Add, you can specify individual group members without bringing along the entire group. However, if you use By List, only groups are listed (as well as other objects), not group members.

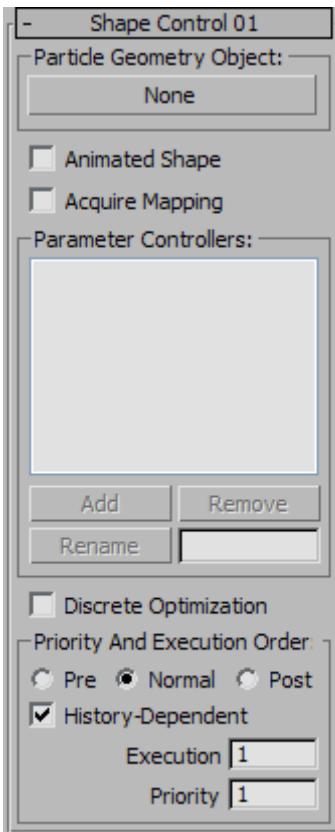
Unlike similar lists in 3ds Max, you can add the same object to this list as many times as you like. This is because the placement in the list of an object determines its index, which other suboperators such as [Geometry](#) can operate on. Adding an object to the list several times can give it extra weight, compared to other objects, for certain calculations.

Shape Control Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Shape Control.

Use Shape Control to assign shapes to particles. It is more flexible than the Shape Instance operator because it allows direct control of the shape parameters for each individual particle. The Shape Control suboperator can see all the shape *and* modifier parameters. The only restriction is that you can have no more than nine input values (Time plus up to eight controller values), and the material should be set by means other than suboperators; for example, by [Material Inheritance](#).

Interface



Particle Geometry Object – Click to select a geometry object to be used as the particle instance shape.

Animated Shape – When on, a Time-type input for the suboperator is available. You can wire time data, and the particle shape is defined by the geometry animation and the time defined for a specific particle. This option is applicable only if the reference geometry is animated.

Acquire Mapping – Similar to the parameter of the same name in the Shape Instance operator.

Parameter Controllers – The list of the controllers for shape parameters (up to eight). Each controller is listed with its data type and name, and creates a separate input for the suboperator to which to wire a data channel.

Use the Add button to add controllers one at a time, and the Remove button to remove a highlighted controller from the list. To rename a controller, highlight it, enter a new name in the text field next to Rename, and then click Rename. The data type is preserved in the list.

Discrete Optimization – When off, each particle's shape is evaluated on an individual basis. In other words, with N particles, the reference geometry is evaluated N times with different parameters; with a large number of particles this can be quite slow. However, if some parameters can take only limited number of values, and number of these values is significantly less than number of particles, it's better to turn this option on. When on, there is a small penalty due to a preliminary analysis of the input wired values to find the input value matches for different particles. On the other hand, this can be faster since the reference geometry is evaluated with fewer calls: only a call for a different input wired value.

To help speed the operation of custom operators and tests that use Shape Control in large-scale particle systems, use the [Discretizator suboperator](#) between Shape Control and its data-input stream.

Priority And Execution Order – Shape Control is an output-type suboperator, and as such, it has the Priority And Execution Order group of parameters. For details, see [Priority and Execution Order](#).

Switch Suboperator

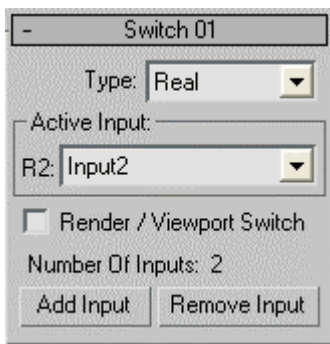
Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Switch.

The Switch suboperator lets you animate switching among several inputs to another suboperator. After adding it, you set the data type, add up to eight inputs, and then use standard 3ds Max keyframing tools (typically Auto Key) to set up an animation so that the active input changes at different times during the animation.

Tip: Animating the Active Input setting creates animation data that is editable in Track View.

Note: The Switch suboperator supports the Object data type, so you can easily switch among different sets of reference objects. Animating the Active Input setting lets you change the reference object during an animation; however, this is not officially supported, and can lead to unexpected behavior.

Interface



Type – Choose the data type input and output by the Switch suboperator from the drop-down list. You can change this before and after adding inputs. Most of the data types are self-explanatory; for explanations of the others, see [Data Types](#).

Active Input – Choose the active input from the drop-down list. If you have Auto Key on, changing the active input creates an animation key at the current frame.

Render/Viewport Switch – Lets you organize two different data flows: one for rendering and the other for the viewports. The check box is available only when Number Of Inputs equals 2. In this case, the first input is equivalent to Render, and the second input is equivalent to Viewport. Choose the input, and then turn on Render/Viewport Switch, whereupon Active Input becomes unavailable and shows the active choice. Also, when Render/Viewport Switch is on, the Add Input and Remove Input buttons become unavailable.

Number Of Inputs – This read-only field shows the current number of inputs. Use Add Input and Remove Input (see following) to change the number of inputs.

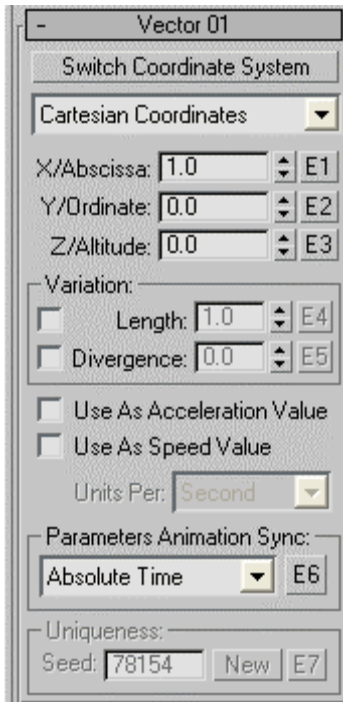
Add Input – Opens the Add Switch Input dialog; enter a name or accept the default, and then click Add or Cancel. Unavailable when eight inputs (the maximum) exist.

Remove Input – Opens the Select Input dialog, with a scrolling list of the current inputs. Highlight an input to delete and then click Remove or Cancel. The removed input no longer appears in the list, and the associated input connector is removed from the suboperator.

Vector Suboperator

Path: Particle View > Data Icon/Operator/Icon Test/Test > Click Edit Data Flow > Add or select Vector.

The Vector suboperator generates vector data; that is, numerical values that can indicate magnitude and direction, or any other data that can be specified by three real numbers. To generate scalar data (magnitude, no direction), use the [Scalar suboperator](#).



Switch Coordinate/Orientation System – Lets the software recalculate settings from one coordinate system to another. Set the coordinate values in one system, and then click the button to get the corresponding values for a different coordinate system.

When the vector type is Cartesian Coordinates, Cylindrical Polar Coordinates, or Spherical Polar Coordinates, clicking this button cycles through those three types and adjusts the values accordingly. When the vector type is Euler Angles, or RPY (Roll, Pitch, and Yaw), clicking this button switches between those two types and recalculates the values. The button is disabled for the other vector types, and when E1, E2, or E3 is on.

[Vector Type] – Choose the type of values the vector output is to represent from the drop-down list. The available vector types are:

- Cartesian Coordinates
- Cylindrical Polar Coordinates
- Euler Angles

- Float Triplet
- Percent Triplet
- Spherical Polar Coordinates
- Roll, Pitch, and Yaw

Immediately below the Vector Type list are three numeric parameters; their labels depend on the active Vector Type choice. You can

E1, E2, E3 – Add [Equal-type](#) data inputs to let the user control the vector values. These can receive input only from a [Parameter suboperator](#) set to an appropriate type. Use [Expose Parameters](#) to make the setting available in the Particle View interface.

Note: Turning on any of these buttons makes the Switch Coordinate/Orientation System button unavailable.

Variation group

Use these settings to add random variation to the vector values. The available parameters depend on the vector type choice:

- **Length, Divergence** – These are available only with Cartesian Coordinates, Cylindrical Polar Coordinates, and Spherical Polar Coordinates; that is, option types that create a vector in 3D space. The Length value specifies the maximum variation in length for the vector value generated. The Divergence value specifies the maximum variation in direction (deviation) of the vector value generated, in degrees.
- **Offset Max** – This is the only variation parameter available with Euler Angles, Roll, Pitch, and Yaw, Float Triplet, and Percent Triplet. It specifies the maximum variation for each component of the vector value generated.

E4, E5 – Add [Equal-type](#) data inputs to let the user control the variation values. These can receive input only from a [Parameter suboperator](#) set to an appropriate type. Use [Expose Parameters](#) to make the setting available in the Particle View interface.

Use As Acceleration Value – When on, the output value is specified as an acceleration value, in units per frame, second, or tick. Available for all vector types. When on, enables the Units Per setting: Choose Frame, Second, or Tick.

Use As Speed Value – When on, the output value is specified as a speed value, in units per frame, second, or tick. Available only when the vector type is Cartesian Coordinates, Cylindrical Polar Coordinates, or Spherical Polar Coordinates. When on, enables the Units Per setting: Choose Frame, Second, or Tick. You can activate only Acceleration Value or Speed Value, not both; clicking again turns the option off.

Parameters Animation Sync – If you animate the suboperator parameters, the software can begin applying this animation to all particles as of the start frame of the animation or the first frame of the current event, or to each particle based on its age. The options are:

- **Absolute Time** – Any keys set for parameters are applied at the actual frames for which they're set.
- **Event Duration** – Any keys set for parameters are applied to each particle relative to the frame at which it first enters the event.
- **Particle Age** – Any keys set for parameters are applied at the corresponding frames of each particle's existence.
- **Particle Lifespan** – Scales/maps the animation of the parameters onto the particle lifespan period. For example, if a parameter value is animated from 5-20 over frames 0-100, then this parameter has the value 5 when the particle is born, and the value 20 when the particle dies. This way you can, for example, define the change in a particle's scale over its lifespan.

For this option to work properly, there must be a Delete operator set to By Particle Age in the flow to define particle lifespan.

- **Input Proxy** – Adds a Time input to the suboperator, to which you can link any other suboperator that outputs data in Time format.

E6 – Adds an [Equal-type](#) data input to let the user control the Animation Sync value. This can receive input only from a [Parameter suboperator](#) set to Type=Animation Sync. Use [Expose Parameters](#) to make the setting available in the Particle View interface.

Uniqueness group

Seed – Lets you set a random seed to differentiate behavior from other randomly seeded functions. Change the Seed value to vary the random variation set in the Variation group, either by entering a value or clicking New. Available with all output types except Boolean.

E7 – When on, you can expose the Seed parameter via a [Parameter suboperator](#) and let the user choose it. Turn on E7, add a Parameter suboperator set to Type: Uniqueness Seed, wire it to the E7 input on the Scalar suboperator, and then use [Expose Parameters](#) to make the setting available in the Particle View interface.

Cache Disk Operator

Path: Particle View > Add or select Cache Disk.

The Cache Disk operator works much like the Cache operator included with the original Particle Flow system, except that it lets you store the cache in a disk file separately from the MAX file. As with the original Cache operator, it lets you pre-calculate and store all the activity in a particle system so that you can play or scrub the animation quickly without having to wait for calculations. The main difference, other than the use of the disk file, is that there is no option for automatic updating, because disk caching is slower than memory caching. The benefit is that the only limit on the size of the cache is the amount of free space on the hard drive.

To use Cache Disk, add it to the event you wish to cache, or to the global event if you want to cache the entire particle flow, click Select File, specify a file name, and then click Save. Then, to create the cache, play the animation or click the Update button.

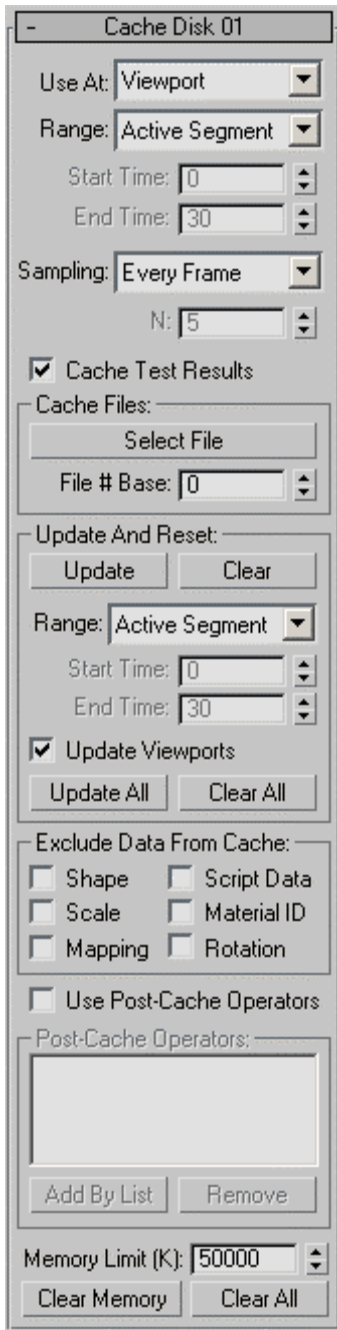
Note: Because disk caching is slower than memory caching, Cache Disk does not have an automatic-update option. If you change a parameter and want to see the changed results, you must click the Update button.

Tip: If you have a situation where particle systems are dependent on each other, use a Cache operator and solve the simulation before you can render, especially if you're rendering over a network, or rendering frames non-sequentially. Since the particle systems depend on each other, Particle Flow can iterate only one frame at a time.

For general background information on how caching works in Particle Flow, see the introduction to the Cache Operator topic in the *3ds Max User Reference*.

Also see [Cache Selective Operator](#).

Interface



Use At – Caches particle motion when playing back in the viewports, or at render time, or both. Default=Viewport.

Important: Choose the Viewport/Render option only when using the same number of particles in the viewports and for rendering; that is, the global event > Emission Rollout > Quantity Multiplier settings must be identical. Otherwise, unpredictable results can occur.

Range – Sets the frame range within which the Cache Disk operator operates. Default=Active Segment.

- **Active Segment** – The software caches only frames in the active segment, as defined by the Start Time and End Time settings on the Time Configuration dialog. This is the frame range shown on the track bar. You can also change the active segment by holding down ALT and CTRL and dragging the track bar with the left, middle, or right mouse button.
- **Custom** – The software caches only frames in the custom range, as defined by the Cache Disk operator's Start Time and End Time settings (see following).

Note: If you cache only part of the animation, Particle Flow calculates particle behavior in subsequent, non-cached frames using the cached data. For example, if you cache frames 0 to 50, and then jump to frame 60, Particle flow will calculate frame 51 based on the cached data, and frames 52 to 60 based on each previous frame.

Start/End Time – The first and last frames of the range considered for caching when Range=Custom (see above). Defaults=0, 30.

Note: The frame range time frame is in absolute time; that is, in terms of the entire animation. If you use a Cache Disk operator locally, and specify a frame range during which no particles are present in the event, Particle Flow won't use the cache.

Sampling – Determines how often the Cache Disk operator samples and caches the animation. Default=Every Frame.

- **Every Frame** – The software caches animation data once per frame.
- **Integration Step** – The software caches animation data at each integration step, using the Integration Step setting as specified by the Use At setting (see above) and on the System Management rollout for the flow (select the global event). If Use At is set to Viewport/Render, it uses lower of the two Integration Step values. For example, if Viewport is set to Half Frame, and Render to 1/8 Frame, the sampling rate would be eight per frame.
- **Every Nth Frame** – The software caches animation data at frame intervals specified by the N value, below.

N – Determines the frame interval for caching when Sampling (above) is set to Every Nth Frame. Default=5.

For example, with N set to the default value of 5, the cache stores animation data for every fifth frame.

Cache Test Results – When caching particle data, this caches the results of test actions as well. Default=on.

This is important if Cache Disk is used as a local operator, and the next event doesn't have a Cache-type operator. For the next event to work properly, it should receive particles from the current event. Those particles result from the activity of a test action. The Cache Disk operator can record the test activity to play it back later.

If the Cache Disk operator is used as a global operator, there is no need to cache the test results. This is because the system has cache data for every event, and is able to jump to an arbitrary frame without the need for test results.

Cache Files group

Each Cache Disk operator stores its data in a sequence of disk files; one for each animation frame. The filename uses the format *base_name#####.pfc*, where *base_name* is the name you supply, *#####* is a four-digit number with leading zeroes, and *.pfc* (Particle Flow Cache) is the filename extension.

[Select File button] – Click this button to specify a location and base name for the cache files. After you do so, the base name appears on the button. You can see the entire path and base name on a tool tip by hovering the mouse cursor over the button.

Update And Reset group

Cache Disk does not have an option to update the disk cache automatically, because disk caching is slower than memory caching. If you change a parameter and want to see the changed results, you must click the Update button.

Update – If you change a parameter in the particle system, the cached data might become invalid. Click this button to update the cache manually, using the range specified in this group.

To cancel the update while in progress, press the Esc key.

Clear – Deletes the cache files, using the range specified in this group.

Range – Sets the frame range within which the Cache operator recalculates data when you click Update or Clear. Default=Active Segment.

- **Active Segment** – The software updates the cache only for frames in the active segment, as defined by the Start Time and End Time settings on the Time Configuration dialog. This is the frame range shown on the track bar. You can also change the active segment by holding down ALT+CTRL and dragging the track bar with the left, middle, or right mouse button.
- **Custom** – The software updates the cache only for only frames in the custom range, as defined by the Start Time and End Time settings (see following).

Start/End Time – The first and last frames of the range that's updated when Range=Custom (see above). Defaults=0, 30.

Update Viewports – When on, the animation plays in the viewports during manual updating of the cache. Turn this off to disable playing the animation in the viewports during manual caching; this can speed up the caching process, especially with large or complex particle systems. Default=on.

Update All – Updates the caches for all Disk Cache operators in the current flow. Use this to avoid having to update each cache individually after changing a parameter that affects the entire flow.

Clear All – Deletes the cache files for all Disk Cache operators in the current flow.

Exclude Data From Cache – Prevents Particle Flow from saving the specified animation data in the cache files. Turn on an option to exclude that type of data from the cache. The data types are: Shape, Script Data, Scale, Material ID, Mapping, and Rotation.

Using these options helps reduce the size of the cache files, especially if the excluded data will be calculated by the post-cache operators (see following). Usually Shape data consumes the most storage. If particle shapes are defined by a post-cache operator then there is no need to keep the data in the cache files, as it will be disregarded anyway.

Use Post-Cache Operators – Lets you apply operators after the caching operation, so their actions are not stored in the cache. After turning this on, click Add By List to open a dialog that lists all behavior-affecting operators in the current flow, and then choose the operators from the dialog. Thereafter they appear in the Post-Cache Operators list.

Use post-cache operators to fine-tune the pre-calculated particle system. Sometime the majority of calculations are spent for particle movement (due to collisions with reference object, and possibly inter-particle collisions). In this case it is best to calculate the motion first and store it in the cache files. The other particle aspects (size, shape, color, mapping, orientation) can be calculated after caching, and applied later.

Post-Cache Operators – Lists the operators specified with Use Post-Cache Operators.

Add By List – Opens a dialog that lets you choose operators whose affects should be applied after the caching operation, so their actions are not stored in the cache.

Remove – Deletes any highlighted operators from the Post-Cache Operators list.

Memory Limit – For optimal performance, the Cache Disk operator keeps part of the cache in memory. This parameter determines the maximum size of the RAM cache. If the total cache size exceeds the Memory Limit value, the operator keeps only the most recent frames in RAM. Once the cache info for a particular frame becomes available, it is written on disk in the form of file with extension .pfc.

Clear Memory – Purges the RAM cache and forces the operator to get the cache data from files (if available) or recalculate it (if cache files are not available).

Clear All – Performs the Clear Memory function (see preceding) for all Cache Disk operators.

Cache Selective Operator

Path: Particle View > Add or select Cache Selective.

The Cache Selective operator works much like the Cache operator included with the original Particle Flow system, except that it lets you exclude certain types of data from the cache. Also, as with the Cache Disk operator, you can specify post-cache operators, and the cache must be updated manually.

The Cache Selective workflow is somewhat different from that of the original Cache operator, in that it's designed to let you define the most calculation-intensive properties of a particle system (usually the motion), pre-calculate it once, and then work with other particle system properties via post-cache operators (shape, size, orientation, mapping, color etc.).

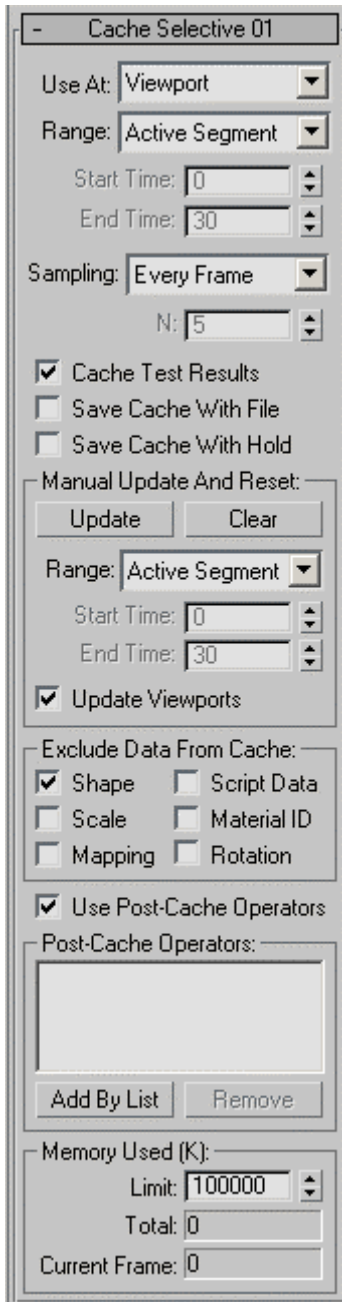
To use Cache Selective, add it to the event you wish to cache, or to the global event if you want to cache the entire particle flow.

Tip: If you have a situation where particle systems are dependent on each other, use a Cache operator and solve the simulation before you can render, especially if you're rendering over a network, or rendering frames nonsequentially. Since the particle systems depend on each other, Particle Flow can iterate only one frame at a time.

For further background information, see the introduction to the Cache Operator topic in the *3ds Max User Reference*.

Also see [Cache Disk Operator](#).

Interface



Use At – Caches particle motion when playing back in the viewports, or at render time, or both. Default=Viewport.

Important: Choose the Viewport/Render option only when using the same number of particles in the viewports and for rendering; that is, the global event > Emission Rollout > Quantity Multiplier settings must be identical. Otherwise, unpredictable results can occur.

Range – Sets the frame range within which the Cache Selective operator operates. Default=Active Segment.

- **Active Segment** – The software caches only frames in the active segment, as defined by the Start Time and End Time settings on the Time Configuration dialog. This is the frame range shown on the track bar. You can also change the active segment by holding down ALT and CTRL and dragging the track bar with the left, middle, or right mouse button.
- **Custom** – The software caches only frames in the custom range, as defined by the Cache Selective operator's Start Time and End Time settings (see following).

Note: If you cache only part of the animation, Particle Flow calculates particle behavior in subsequent, non-cached frames using the cached data. For example, if you cache frames 0 to 50, and then jump to frame 60, Particle flow will calculate frame 51 based on the cached data, and frames 52 to 60 based on each previous frame.

Start/End Time – The first and last frames of the range considered for caching when Range=Custom (see above). Defaults=0, 30.

Note: The frame range time frame is in absolute time; that is, in terms of the entire animation. If you use a Cache Disk operator locally, and specify a frame range during which no particles are present in the event, Particle Flow won't use the cache.

Sampling – Determines how often the Cache Disk operator samples and caches the animation. Default=Every Frame.

- **Every Frame** – The software caches animation data once per frame.
- **Integration Step** – The software caches animation data at each integration step, using the Integration Step setting as specified by the Use At setting (see above) and on the System Management rollout for the flow (select the global event). If Use At is set to Viewport/Render, it uses lower of the two Integration Step values. For example, if Viewport is set to Half Frame, and Render to 1/8 Frame, the sampling rate would be eight per frame.
- **Every Nth Frame** – The software caches animation data at frame intervals specified by the N value, below.

N – Determines the frame interval for caching when Sampling (above) is set to Every Nth Frame. Default=5.

For example, with N set to the default value of 5, the cache stores animation data for every fifth frame.

Cache Test Results – When caching particle data, this caches the results of test actions as well. Default=on.

This is important if Cache Disk is used as a local operator, and the next event doesn't have a Cache-type operator. For the next event to work properly, it should receive particles from the current event. Those particles result from the activity of a test action. The Cache Disk operator can record the test activity to play it back later.

If the Cache Disk operator is used as a global operator, there is no need to cache the test results. This is because the system has cache data for every event, and is able to jump to an arbitrary frame without the need for test results.

Save Cache with File – When on, the software includes the cached data with scenes that you save to disk. This can significantly increase the size of saved files, but saves the time of recalculating the particle motion upon reloading the file. Default=off.

Normally, the cache data is saved only in disk files that you create with the Save or Save As commands. You can also instruct the software to include cached data in held files using the following option.

Save Cache with Hold – Saves cached data in the Hold file, created with 3ds Max > Edit menu > Hold. Default=off.

Update And Reset group

Cache Selective does not have an option to update the disk cache automatically, because this can negatively impact performance. If you change a parameter and want to see the changed results, you must click the Update button.

Update – If you change a parameter in the particle system, the cached data might become invalid. Click this button to update the cache manually, using the range specified in this group.

To cancel the update while in progress, press the Esc key.

Clear – Deletes the cache files, using the range specified in this group.

Range – Sets the frame range within which the Cache operator recalculates data when you click Update or Clear. Default=Active Segment.

- **Active Segment** – The software updates the cache only for frames in the active segment, as defined by the Start Time and End Time settings on the Time Configuration dialog. This is the frame range shown on the track bar. You can also change the active segment by holding down ALT+CTRL and dragging the track bar with the left, middle, or right mouse button.
- **Custom** – The software updates the cache only for only frames in the custom range, as defined by the Start Time and End Time settings (see following).

Start/End Time – The first and last frames of the range that's updated when Range=Custom (see above). Defaults=0, 30.

Update Viewports – When on, the animation plays in the viewports during manual updating of the cache. Turn this off to disable playing the animation in the viewports during manual caching; this can speed up the caching process, especially with large or complex particle systems. Default=on.

Update All – Updates the caches for all Disk Cache operators in the current flow. Use this to avoid having to update each cache individually after changing a parameter that affects the entire flow.

Clear All – Deletes the cache files for all Disk Cache operators in the current flow.

Exclude Data From Cache – Prevents Particle Flow from saving the specified animation data in the cache files. Turn on an option to exclude that type of data from the cache. The data types are: Shape, Script Data, Scale, Material ID, Mapping, and Rotation.

Use Post-Cache Operators – Lets you apply operators after the caching operation, so their actions are not stored in the cache. After turning this on, click Add By List to open a dialog that lists all behavior-affecting operators in the current flow, and then choose the operators from the dialog. Thereafter they appear in the Post-Cache Operators list.

Post-Cache Operators – Lists the operators specified with Use Post-Cache Operators.

Add By List – Opens a dialog that lets you choose operators whose affects should be applied after the caching operation, so their actions are not stored in the cache.

Remove – Deletes any highlighted operators from the Post-Cache Operators list.

Memory Used (K) group

The Cache Selective operator stores data in system memory; you can specify an upper limit for the amount of memory it uses. If the Limit setting and the amount of cached data exceeds the available free memory, the computer system might use virtual (hard disk-based) memory instead, which slows down the caching. If Particle Flow fills the cache, any remaining frames are calculated on the fly.

This group also lets you monitor the amount of memory used for caching data.

Limit – The maximum amount of system memory used to cache particle data, in kilobytes. Default=100,000, or 97.6 MB.

Total – The amount of memory currently used by the cached data, in kilobytes. Read-only.

Note: Even animation frames with no particles will probably consume a certain amount of cache memory. The reason for this is that the cache also stores states for randomly calculated values such as Variation, to ensure that particle activity is consistent across a rendering network, and with machines that might not have regular access to all frames.

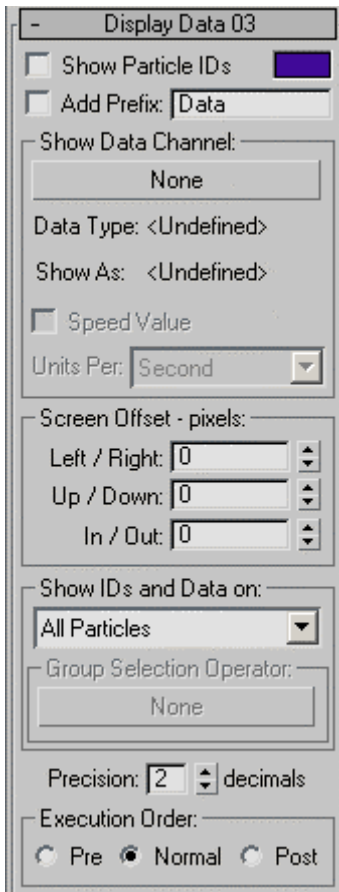
Current Frame – The amount of memory used by the data cached for the current frame, in kilobytes. Read-only.

Display Data Operator

Path: Particle View > Add or select Display Data.

The Display Data operator shows, in the viewport, numerical data created by a Data operator or test for certain or all particles in the event containing the Display Data operator, or, if the operator is in the global event, particles in the flow.

Interface



Show Particle IDs – Displays each particle's ID number before the displayed data. Particles are numbered in the order of their birth, starting with 1 for the first particle born. The ID number is followed by a > symbol to distinguish it from the data.

[color swatch] - Shows the color used to display the data. The software chooses a different color at random for each Display Data operator added to the system. To change the color, click the color swatch and use the Color Selector dialog to choose a new color.

Note: If the emitter is selected, the data is displayed in white.

Add Prefix – Lets you add a custom prefix, which you enter in the text field. It's not necessary to append the text with a space; the operator automatically offsets the prefix from the data. The prefix is displayed between the ID (if shown) and the data.

Show Data Channel group

[Data Channel button] – Click this button to choose the channel for the data to display. Choose the channel from the Select Data Channel dialog.

Data Type – This read-only field shows the type of data that is being displayed.

Show As – Shows how the data is being displayed. If only one option is available, the field is read-only. If several options are available, such as in the case of Real data, a drop-down list lets you choose an option.

Spin Rate Value/Speed Value – If Show As is set to Angle, this option becomes available as Spin Rate Value. If Show As is set to World Unit, this option is available as Speed Value. Turn on to enable the option.

Units Per – When Spin Rate Value/Speed Value is on, use this setting to choose the time frame for the spin rate or speed value.

Screen Offset group - Use these values to adjust the positioning of the script data relative to the particle position, in pixels. By using different offset values and several Script and Display Script operators, you can show multiple data at once.

Show IDs and Data on – Lets you filter the particles for data display. The options are:

- **All Particles** - Displays data for all particles in the viewports.
- **Selected Particles Only** - Displays data only for selected particles. To select particles, use the Modify panel > Selection rollout controls and use standard viewport selection methods to pick particles to see their data (see the Particle Flow documentation for further information).
- **Group Selected Only** - Displays data only for group-selected particles. Use the Group Selection Operator button to specify the applicable operator. This option is applicable only if you have *Particle Flow Tools: Box#1* installed (commercial or demo version; the latter is available from www.orbaz.com/download). It can be used in conjunction with a Camera Culling operator from *Particle Flow Tools: Freebies* (also available from www.orbaz.com/download), which can serve as a Group Selection operator as well. In this case, you can show only data for particles that are visible to the camera.

Group Selection Operator – If Particle Flow Tools: Box#1 is installed and you've set Show IDs And Data On to Group Selection Only and your particle system includes at least one Group Selection operator, click this button to specify the Group Selection or

Camera Culling operator to whose member particles the Display Data operator should apply.

Precision – For float values, sets the number of decimal places displayed. Range=0 to 8 (where 0 is no decimal places).

Execution Order – Sets the timing for the data display:

- **Pre** – Shows particle data values before any operator in the event is executed.
- **Normal** – The data are shown as of the current execution order of the operators. You can use two Display Data operators--above and below the Data operator in question--to see how the Data operator changes some data.
- **Post** – Data values shown after all the operators in the event are executed.

Updated Particle Flow Operators

Path: Particle View > Add or select Shape Instance or Age Test operator.

Particle Flow Tools: Box#3 provides updated versions of the Shape Instance and Age Test operators, both originally part of Particle Flow as included with 3ds Max. This topic describes only the new parameters for these operators.

Shape Instance

Fast Shape Evaluation – When on, this option speeds up execution of the Shape Instance operator by evaluating the particle shape at the final integration step only. This is similar to History-Dependent option for Output suboperators. You can see how the option works with the example scene SpidersForCaching.max. Compare the time to switch from first frame to the last frame in viewport with this option on and off.

Age Test

Adjustable Age – Script and Data operators can modify the age of particles on the fly. However, for performance purposes the standard Age Test operator caches each particle's when it first enters the event, and then uses the cached value for testing. Modifying the age the particle enters the event can create test discrepancies. This new option resolves this issue. If the event contains Script or Data operators that modify particle age or event age, turn Adjustable Age on.

Material Inheritance

Path: Click the title bar of any non-global event in Particle View > Event ## rollout

Particle Flow imposes a specific limitation on material usage in events. In particular, a material assigned to an event—either via any standard Material operator or as a parameter of the standard Shape Instance operator—does not travel along with the particles from one event to the next. This diminishes the overall flexibility of a Particle Flow system, and necessitates using some workarounds: Either assign a Material operator to the global event, or assign a Material operator to all affected events.

Box#3 enhances event functionality to improve the overall handling of the materials in a PFlow system. It still does not allow particles to carry a material between events, but it provides an assortment of tools that simplify material assignment and inheritance.

In the context of material assignment to Particle Flow events, a material can be categorized as either *parametric* or *synthetic*.

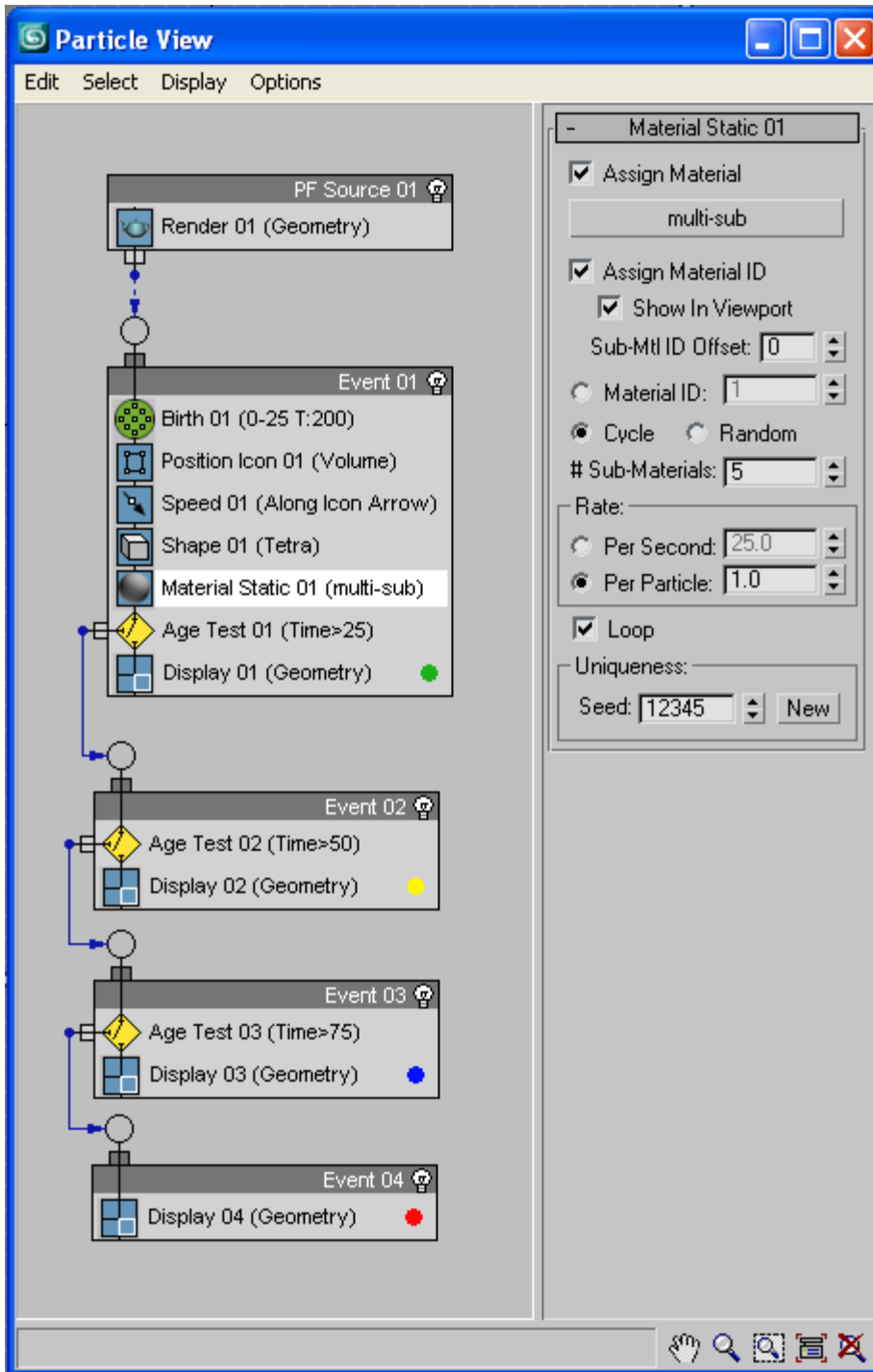
A material is parametric when you use any of the standard Material operators. The operator has a button to define a material from the Material Editor. This material is passed as a reference to the event containing the Material operator to be used by the particles while they reside in this event. Again, this is a *parametric* material; it's assigned as a parameter in a Material operator.

A material is synthetic when you use, for example, the standard Shape Instance operator. When the operator's Acquire Material is on, it takes the material assigned to the reference geometry and assigns it to the current event. Furthermore, if the operator refers to a group of objects as a Particle Geometry Object, the operator gathers all materials from the group, and creates a new Multi/Sub-Object material to be assigned to the event. In this situation, a new material is *synthesized* from other materials.

If you click the title area of a global event, you can see the parameters relevant to the whole PFlow particle system. Now, with Box#3 v1.5, you can also click the title area of a local event and see a few tools related to material handling, and also the material assigned to the event.

Open the included simple example file *MaterialInheritance01.max* to see how the new Material tools work. The scene has a simple Particle Flow system with four events. It contains no Material operators; the colors of the particles in the viewport are defined by a different Display operator in each event, so that when you play the animation you can tell by their color change when the particles switch events.

Now, add a Material Static operator to Event 01, select it, and assign the “multi-sub” material from the Material Editor (the first slot) as an instance (the default mode), and adjust the Material Static parameters to cycle through the sub-materials. Your event map will look like this:

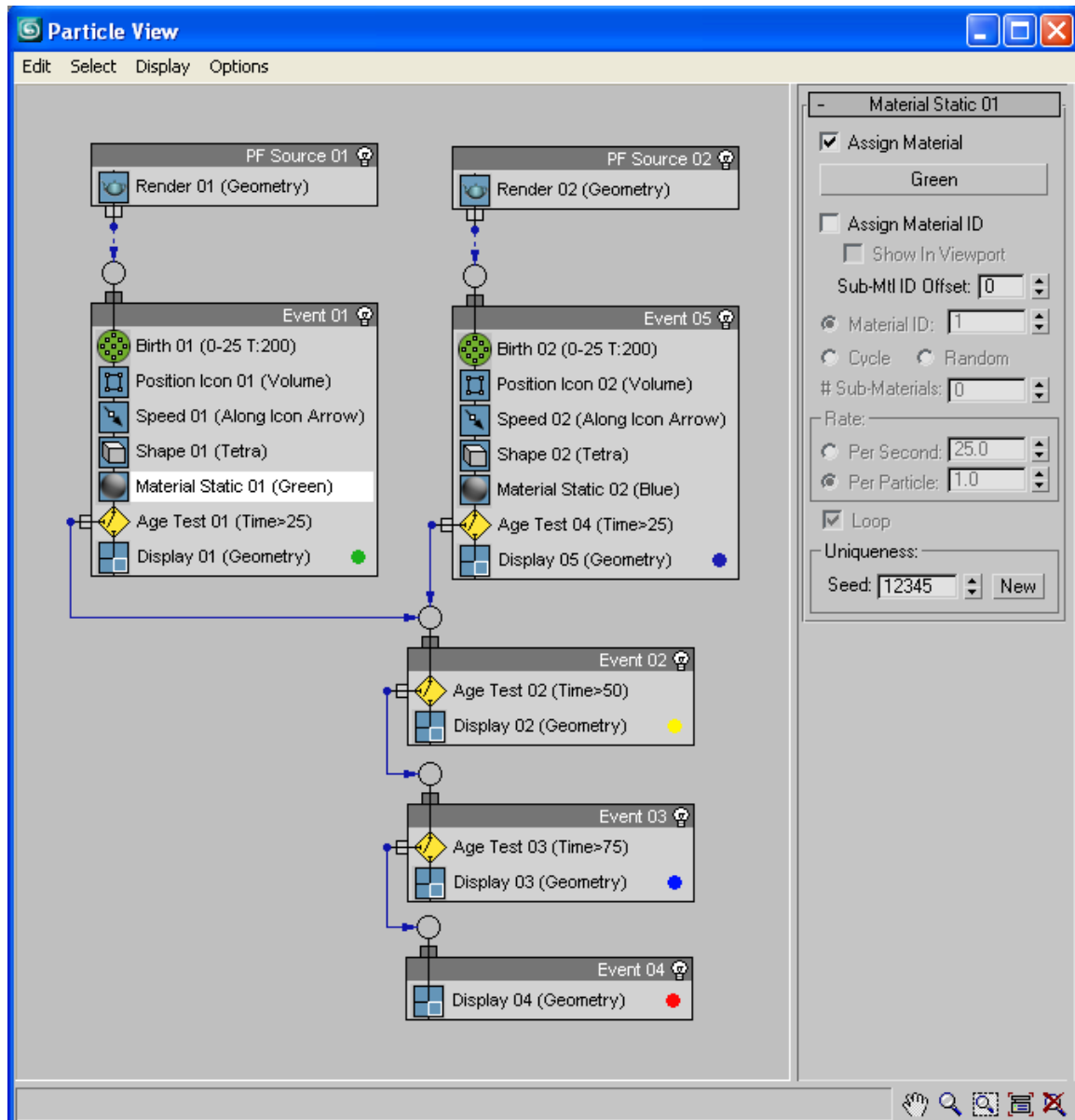


If you play the animation now, you see that in the beginning, particles all have different colors, but then they switch, as a block, to a different color. This is due to the fact that the next three events have no Material operator, and thus use the Display color in the viewports.

Click the title bar of Event 01 to show the parameters of Event 01, and then click the Downstream Materials button. Particle Flow reports that it has updated three events with

a material. Play the animation again; the particle colors are now consistent in all events. Also, if you select the other events, you can observe that now the Current Material button indicates the same material as in the first event.

Next, look at the more complex example *MaterialInheritance02.max*, also included with Box#3. It has two streams of particles converging to a common stream. The start events have different materials defined by Material Static operators. Once the particles come to the second event (Event 02), they lose their materials. Let's see what we can do in this situation.



Select Event 01 and click the Downstream Materials button. You can examine the material assigned to events Event 02, Event 03, and Event 04, simply play the animation; the result is that all events downstream from the Event 01 inherited the Green material.

That does not work well because particles from Event 05 got the Green material as well. We need to fix this situation somehow. The goal here is to allow the particles that originate in the two birth events to keep their original respective materials.

First, try the following: select Event 05 (the start event with the Blue material), and click Downstream Materials button. You might remember that when you clicked the button in Event 01, the software reported “Materials in 3 Events were updated.” Now, with Event 05, the report is different: “All Events have valid materials.” That tells you that the materials in the downstream events were not changed. If you play the animation or examine the downstream events, you can see that they still hold the Green material from Event 01. Why?

When you click the Downstream Materials button, the plug-in finds all events downstream from the current event, and for each downstream event it triggers Material Update. A material in an event is updated according to its Material Inheritance parameters. If the Inherit type is None, or the event has its own operator that holds a material (any Material operator, or a Shape Instance operator with material acquisition), then the event does not update its material because the material in the event is already defined. If Inherit type is set to Highest Priority then the event looks for its upstream events (Event 02’s upstream events are Event 01 and Event 05), checks their Material Priority values, and inherits a material from the event with the highest priority. In this case, both Event 01 and Event 05 have the same Material Priority: 0 (zero). So the result is ambiguous: Event 02 can choose the first upstream event that it finds. To enforce the material inheritance from Event 05, change Material Priority value in Event 05 to 1, and click Downstream Materials button again. Now you can see that the downstream events have the Blue material.

The stated goal remains elusive: The particles from Event 01 change their material from Green to Blue once they come to Event 02. So we need to do something else.

The last option for the Inherit type is Combine Materials. That's the type to use in order to propagate materials properly downstream. Select Event 02 and set Inherit to Combine Materials. Now, click the Current Event Material button to pull a combined material from upstream. Now, if you look at the Current Material in Event 02, it's not the Green or Blue material, but a new material labeled something like Material #10. If you drag the material to the Material Editor, you can see that it's a Multi/Sub-Object material with a single sub-material named “Blue.”

Why? Both events Event 01 and Event 05 have a standard material, with Event 05 having a higher priority according to the event settings. Since both these materials are standard materials (as opposed to multi/sub materials), both these materials try to allocate the first slot in the *synthetic* multi-sub material for Event 02. But Event 05 prevails because of the higher priority value.

To resolve the material collision, it's necessary to adjust either of the materials in Event 01 and Event 05. Here's how: Select the Material Static operator in Event 05, and change

the Sub-Mtl ID Offset value to 1 (from the default value 0). Observe that Assign Material ID option is turned on automatically (for a reason to be explained later), but you also need to turn on Show In Viewport in order to see the material effect in the viewport. Also note that the dynamic name of the operator has changed from Material Static 02 (Blue) to Material Static 02 (Blue >> 1), which indicates that the ID offset has been applied.

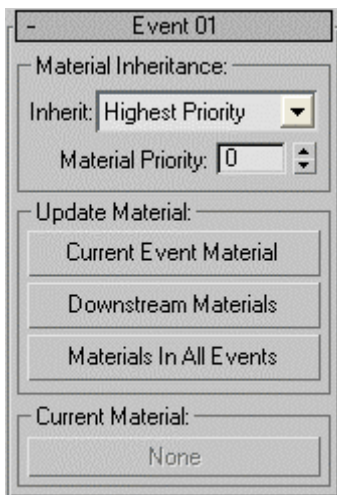
Select Event 05 and examine the material in the Current Material button (drag to the Material Editor). The material is a true Multi/Sub-Object material with two sub-materials: The first slot is empty, and the second slot occupied by the Blue material. It's now possible to combine materials from Event 01 and Event 05 without a collision.

Select Event 02 and click the Update Current Material button. Examine the material in the Current Material button: The combined material has two sub-materials, Green and Blue. Click the Downstream Materials button to propagate this material to events Event 03 and Event 04.

If you play the animation now, you still can see a discrepancy with the particles originated from Event 01: They turn blue once they leave Event 01. To fix that, you need to turn On Assign Material ID and Show In Viewport in Material Static 01 (Green). Even though the Green material does not have sub-materials, the sub-material indices are needed because later on particles leave the event and become governed by the Multi/Sub-Object materials.

Interface

This section describes the material-inheritance controls that you access by clicking an event title bar as well as an additional parameter, Sub-Mtl ID Offset, that has been added (as of Box#3 v1.5) to operators that use materials.



Material Inheritance – Defines how a material is assigned to a current event on the basis of the upstream events, and how the current event can influence the materials downstream.

Inherit:

- **None** – Does not inherit materials from the upstream events.
- **Highest Priority** – From all immediate upstream events, chooses the one with the highest priority, and gets materials from this event.
- **Combine Materials** – Creates a synthetic Multi/Sub-Object material from the materials in the upstream events. If there is a collision between different materials and sub-materials, the entity with the highest priority prevails.

Material Priority – Sets the priority value used by the Highest Priority and Combine Materials options for Inherit (see preceding).

Note: if an event contains a Material-type operator, then the event does not inherit materials from the upstream events; the Inherit type option does not matter.

Update Materials group

This group contains tools to set materials in Particle Flow events.

Current Event Material – Updates the current event material by pulling materials from the upstream events.

Downstream Materials – Updates the materials in all downstream events.

Materials In All Events - Updates materials in all events in all PFlow systems in the scene.

Current Material – Shows the material used by the event. This is a read-only property; you can drag a material from this button to the Material Editor for examination, but you cannot drop a material from Material Editor to this button.

Additional Parameter: Material/Birth Group/Shape Instance operators

Sub-Mtl ID Offset – When using material inheritance with a Multi/Sub-Object material, the Material operators, Birth Group operator, and Shape Instance operator add this value to the Material ID value to create a sub-material ID used by Particle Flow to determine which of the material's sub-materials to inherit and thus apply to the particles in its event, based on the sub-material ID in the material.

Customizing Particle Flow Tools: Box#3 Pro

The 3ds Max system path for plug-in configuration files (typically something like *C:\Documents and Settings\[user name]\Local Settings\Application Data\Autodesk\3dsmax\2009 - 32bit\enu\plugcfg*) contains a initialization (INI) file named *ParticleFlowToolsBox3.ini*. This is a simple XML file that defines default parameter values in Box#3.

After installation this file looks like this:

```
<DefaultSettings>
  <DataViewLayout Width="751" Height="620" DividerHeight="110" />
  <DataViewDisplay ShowParameters="1" ShowDepot="1"
DescriptionType="1" />
  <DataViewOptions UseDynamicNamesForNew="1"
AutoUpdateOnClose="1" />
</DefaultSettings>
<Presets>
  <DefaultReadWrite Path="C:\Documents and Settings\Administrator\Local
Settings\Application Data\Autodesk\3dsmax\2009 - 32bit\enu\plugcfg\Particle
Flow Data Presets\" />
  <AdditionalRead1 Path="D:\3dsMax2009\plugcfg\Particle Flow Data
Presets\" />
</Presets>
```

In this default state, the file overrides the Data View layout and options parameters and it also defines the paths for data presets.

You can modify the default Data View parameters in two ways:

- by adjusting Data View parameters while using Data View in 3ds Max, and then choosing Data View > Options > Save Preferences, or
- by directly adjusting the INI file.

The Presets part of the INI file defines a path where data presets are stored when you create a new data preset (*DefaultReadWrite* tag); and it also defines additional paths to read data presets. You can have up to nine additional preset folders, with the tags *AdditionalRead1*, *AdditionalRead2*, etc.

You can also use the INI file to define default values for Box#3 operators and suboperators parameters.

To override additional parameters, follow this procedure:

1. Open the MAXScript Listener and make sure MacroRecorder > Enable is on.
2. Create a Box#3 PFlow operator whose parameter you want to override and note the name shown in Maxscript Listener for this operator
3. Adjust the parameter that you want to change to the desired value and note how it is shown in Maxscript Listener.
4. Add the data to the INI file using the same form as the records already in the file. Don't forget to put the value in quotes (" ").

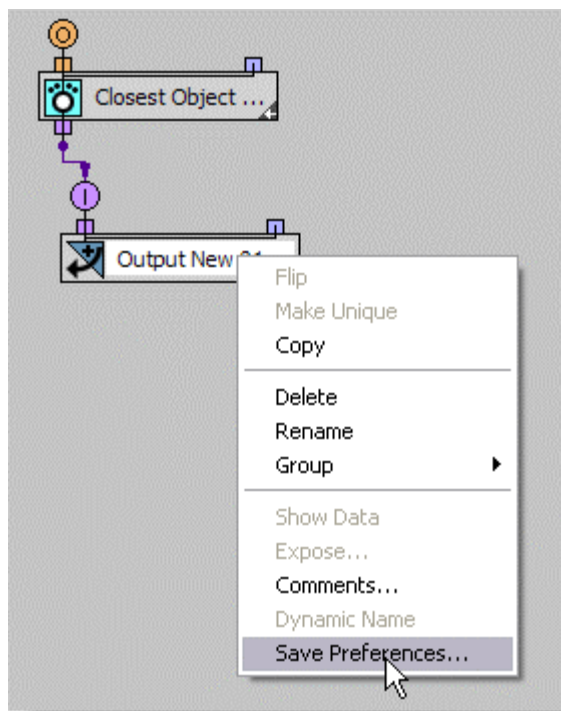
For example, to make the Data Operator > Auto Update setting on by default, add this line in the *DefaultSettings* block:

```
<Data_Operator Auto_Update="on" />
```

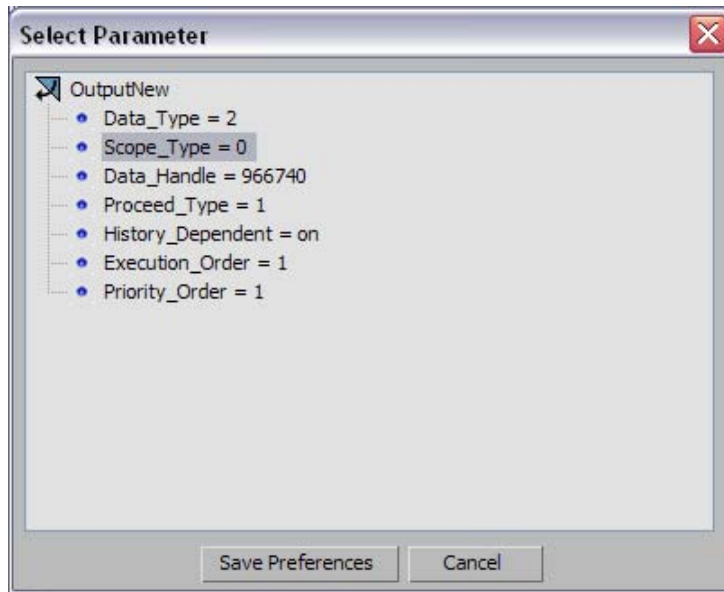
5. Quit 3ds Max and then restart.

For a Box#3 suboperator, a different approach is necessary:

1. Open Data View and create a suboperator whose parameter you want to override.
2. Right-click the suboperator and choose the Save Preferences menu item.



3. In the list of the available parameters, highlight the parameter to override and click Save Preferences.



Keep in mind that the labeling of the parameters listed can differ from the suboperator's UI due to the fact that a suboperator may have a complex system of hiding and enabling different UI configurations, depending on the parameters chosen. The Select Parameter dialog lists all parameters available for overriding. To make sure that you override the proper parameter, change the parameter value in the suboperator UI and verify that its value has also changed on the Select Parameter dialog.

The plug-in reads the content of the Box#3 INI file only when 3ds Max starts, and then uses this information as a possible default override.

After adding overrides for a Data Operator and Amount Change suboperator the INI file can look like this:

Code:

```
<DefaultSettings>
  <DataViewLayout Width="751" Height="620" DividerHeight="110" />
  <DataViewDisplay ShowParameters="1" ShowDepot="1" DescriptionType="1" />
  <DataViewOptions UseDynamicNamesForNew="1" AutoUpdateOnClose="1" />
  <Data_Operator Auto_Update="on" />
  <AmountChange Sub-Type="1" />
</DefaultSettings>
<Presets>
  <DefaultReadWrite Path="C:\Documents and Settings\Administrator\Local
Settings\Application Data\Autodesk\3dsmax\2009 - 32bit\enu\plugcfg\Particle Flow Data
Presets\" />
  <AdditionalRead1 Path="D:\3dsMax2009\plugcfg\Particle Flow Data Presets\" />
</Presets>
```

Note: The folder where the INI file resides depends on various factors and your 3ds Max settings. Typically it's the `\plugcfg\` folder within the path specified by the Customize menu > Configure System Paths > System panel > MaxData setting.