

# Houdini 16 VEX

[Introduction](#)

[Create](#)

[Export](#)

[Solve](#)

[Inputs and Outputs](#)

[Attributes](#)

[Parameters](#)

[Relative References](#)

[Connections](#)

[Common Nodes](#)

[Arithmetic/Aggregation](#)

[Transform](#)

[Noise](#)

[Conversions](#)

[Utility](#)

## Introduction

VEX is a proprietary programming system that's used for a bunch of different things within Houdini: shaders, deforming geometry, particle movements, etc..

Some facts about VEX...

- VEX is inspired by RenderMan's shading language
- VEX has its own compiler (called vcc)
- VEX can be "written" with nodes through VOPs (Vex OPerators)
- Prior to VEX, Houdini provided a scripting system called Hscript

This document will focus on using VEX through the VOPs interface in Houdini. That means that we'll be programming by dragging-and-dropping nodes and connecting them up. We won't writing programs in VEX.

For more information on writing VEX, see the VEX language reference...

<http://www.sidefx.com/docs/houdini/vex/lang>

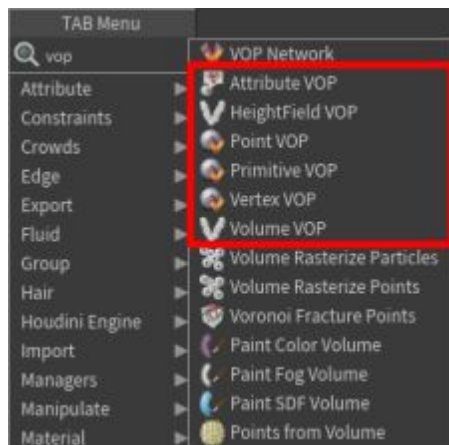
For more of an introduction on VEX/VOPs, see  
[http://www.tokeru.com/cgwiki/index.php?title=Houdini\\_Vops&oldid=2317](http://www.tokeru.com/cgwiki/index.php?title=Houdini_Vops&oldid=2317)

## Create

It looks like you can create VOP nodes in pretty much any context other than directly in /obj...

In the network view, go to the Tab Menu and search for “VOP”. You’ll get a list of VOP node types that you can use.

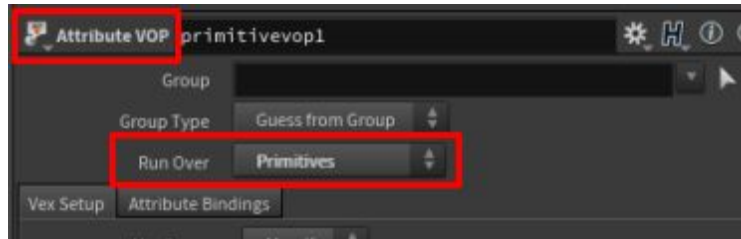
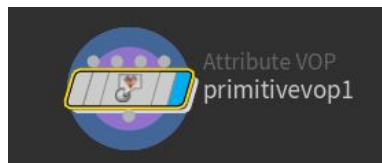
For example, if you go inside one of your geometry nodes and use the Tab Menu to search for VOP, here’s what you’d see...



**NOTE:** You can also add in a node where you directly type in VEX script via the VEX node...



Pretty much all the VOP nodes listed above either boil down to being Volume VOP or Attribute VOP. For example, if you choose Primitive VOP, you’ll actually get an Attribute VOP node but the Run Over property will automatically be set to Primitives.



**NOTE:** Remember that “primitive” is Houdini’s term for polygons/faces.

In the case of Attribute VOP, the Run Over property defines what level of geometry that the VEX operations will be processing. Your VEX operations will update/add/delete attributes on that geometry based on however you defined your logic/operations.

Remember that most of Houdini is about attributes being updated/added/deleted as they flow through the system. For example, if you’re working with points, your points may have an attribute called...

- P → a vector that describes their location in space
- N → a vector that describes their normal (if they have one)
- Cd → a 3D vector used for RGB coloring (if they’re colored)
- uv → a 2D vector that describes the UV coordinate for the point (if mapped to UV)
- etc..

**NOTE:** See <http://www.sidefx.com/docs/houdini/model/attributes> for more common attributes

In addition to these common attributes, you may have attributes that get used by solvers to do computations on the points (e.g. an attribute called v for velocity) or custom attributes that you add yourself (e.g. maybe the artists need attributes to do something or some other Houdini person will use them to do some additional calculations).

## Export

You can export your Attribute VOP node as a Houdini Digital Asset by first putting it in its own Subnetwork and then exporting that subnetwork as a Houdini Digital Asset.

See the Subnetwork section of the main Houdini document to see how to do this.

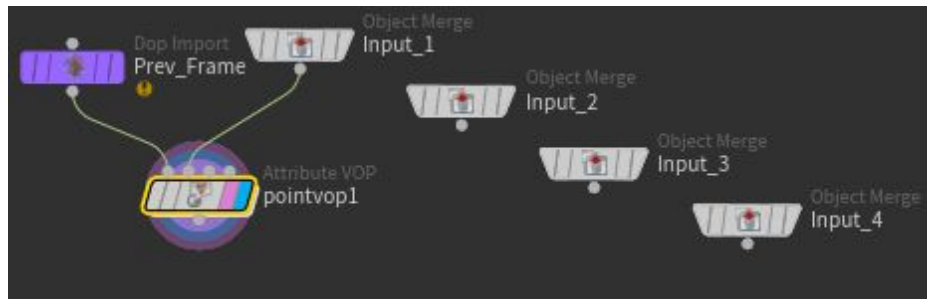
**NOTE:** It looks like previous versions of Houdini let you do this straight from the VOP node (instead of having to create a subnetwork). The option seems to be missing now.

## Solve

In certain cases, you will want the computation happening in your Attribute VOP node to make use of previous frame's geometry. You can do this by putting your Attribute VOP inside of a Solver node...



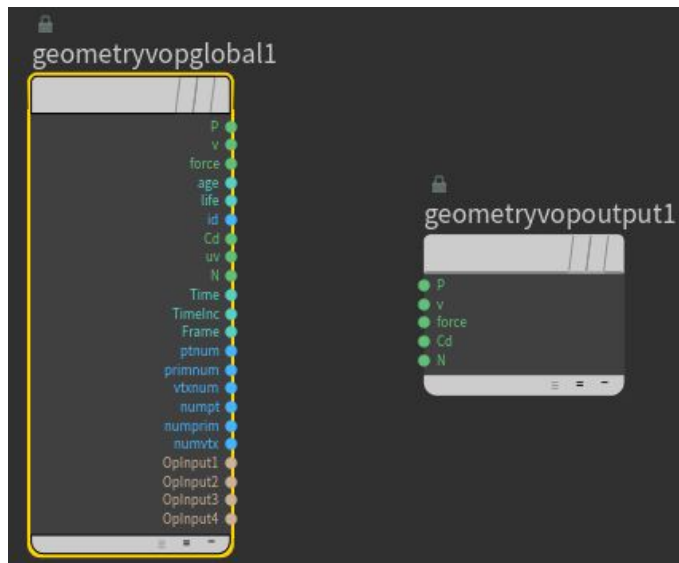
Inside of your Solver, you'll have access to the previous frame's geometry as well as any inputs you fed into the solver node. Simply feed your previous geometry and your current geometry into your VOP node and do whatever computations you need to...



## Inputs and Outputs

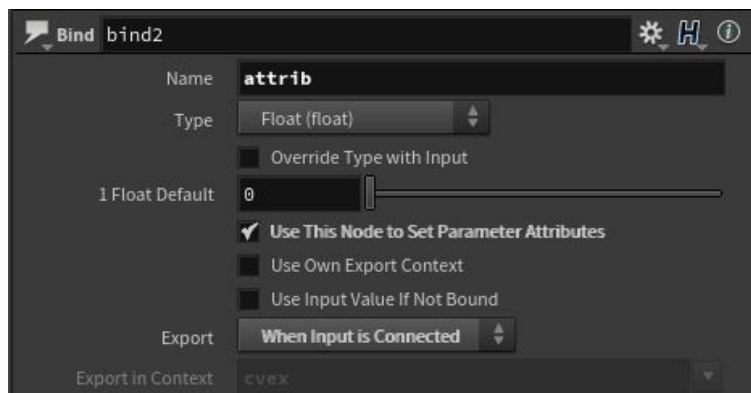
### Attributes

By default, when you create an Attribute VOP node, you'll get 2 nodes that will let you import/export common attributes...



You can drop in a Bind node for any additional attributes that you want to import/export. Just make sure that you...

1. set the Name property to the name of the attribute
2. set the Type property to the correct type the attribute is set to
3. set the Export property to When Input is Connected (the Bind node will gain an input)



**NOTE:** Notice how we dropped 2 Bind nodes for the same attribute (named attrib in the example). If you have multiple Bind nodes pointing to the same attribute, you can only set the parameter on the first Bind node. All other Bind nodes for the attribute will copy those parameters.

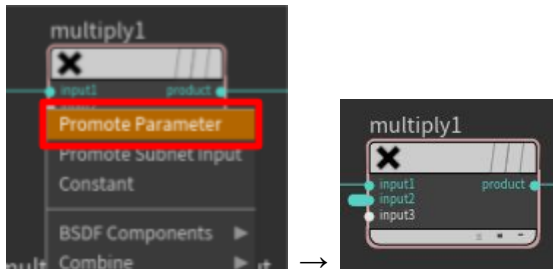
If you want to take the parameters from a Bind node that isn't the first for the attribute, check the 'Use This Node to Set Parameter Attributes' checkbox.

**NOTE:** If you set the attribute to one that's already being imported/exported by the 2 default nodes, your properties are going to be disabled. That's because the 2 default nodes are internally doing the same thing to give you access to these attributes, and they already have the properties set.

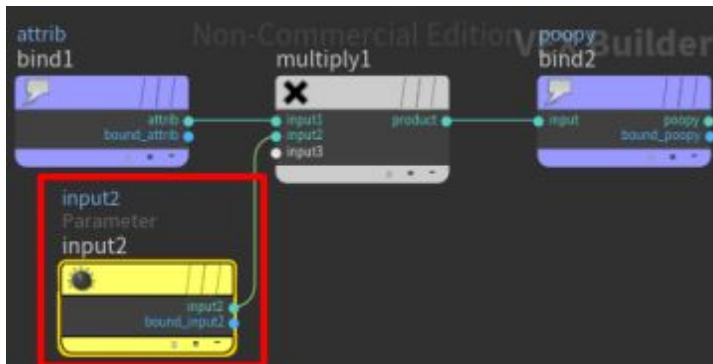
## Parameters

You can expose the inputs of a node my MMB as a Parameter on the Attribute VOP node that it sits in.

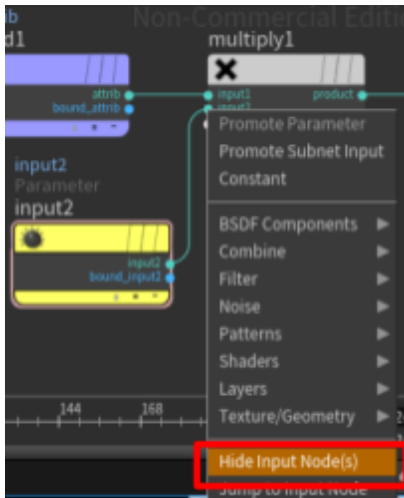
To do this, begin by MMB clicking on the input and choosing Promote Parameter. When you do this, a little nub will appear for that input...



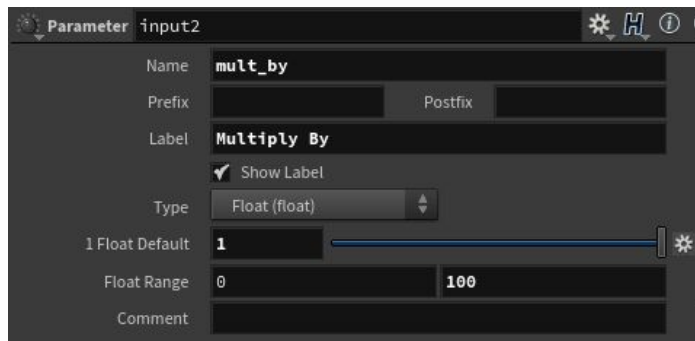
Double-click on the nub to expand out the hidden Parameter node.



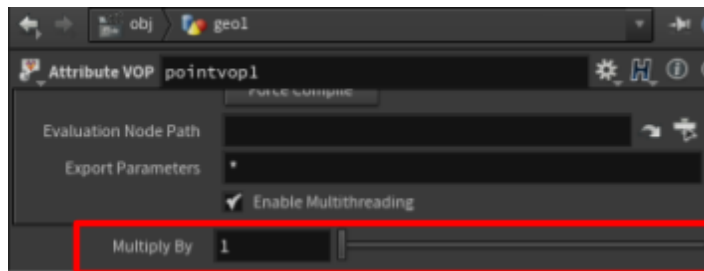
**NOTE:** If you want to turn this back into a nub, MMB click on the input and choose Hide Input Nodes...



In the Parameter node, you can set the name of the parameter, the label that shows up for the parameter, the default value of the parameter, and the range of inputs for the parameter (will show up as a slider)...



If you go back out to your Attribute VOP node, you'll see your new parameter show up in the parameter window...



You can re-organize your Attribute VOP node's parameters via the Edit Parameter Interface. For more information on this, see the Subnetworks section in the main Houdini document.

**NOTE:** We dealt with a basic parameter node, but you also have the option of using a Ramp Parameter node. A Ramp Parameter node makes in some input and changes it so that it fits some ramp property on your Attribute VOP node (it'll show up as an actual ramp vs just a field).



A couple of things to be aware of...

1. the Ramp Type option should almost always be set to Spline Ramp.
2. your input has to be normalized to be between 0 and 1 (I think you can use a Fit Range node to normalize to a 0 to 1 range before feeding in)

## Relative References

You can make a relative reference to the node of any other parameter, even if it's outside of the Attribute VOP node that you're working in...



In the example above, we're going outside of our Attribute VOP node and instead accessing the tx parameter of a node called box1. This is perfectly legal to do.

For more information on setting relative references, see the Parameter View section in the main Houdini document.

**NOTE:** If you want to reference an attribute in the expression, you can do so by prefixing the attribute's name with a @. See

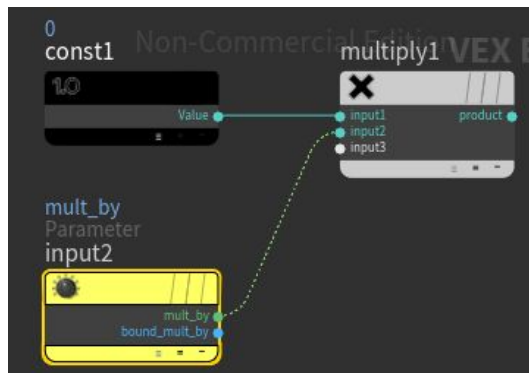
<https://houdinitricks.com/quicktip-new-attribute-syntax-in-houdini-15/>

## Connections

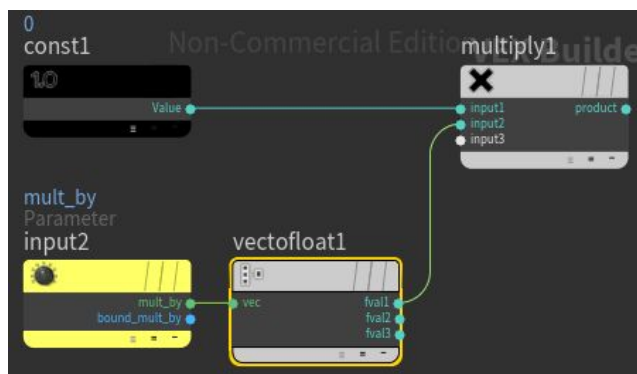
If the connect is a solid line, you're fine.



If the connection shows up as a dotted line, it means that there's some implicit conversion happening between the output and the input. For example, if the node you're working with expects a float but you're giving it a vector of floats, the line will show up as dotted...

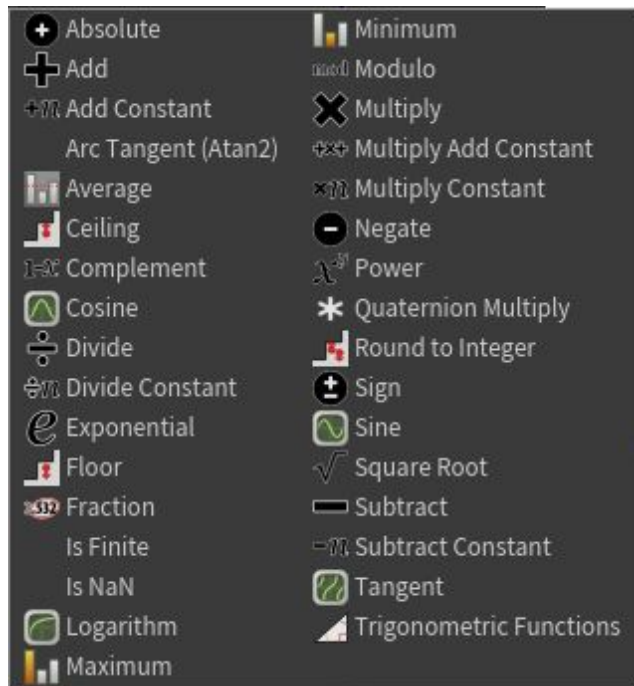


It's best to explicitly convert the data to the type expected rather than relying on these implicit conversions. In the example above, we can convert our vector to a float using a Vector to Float node...



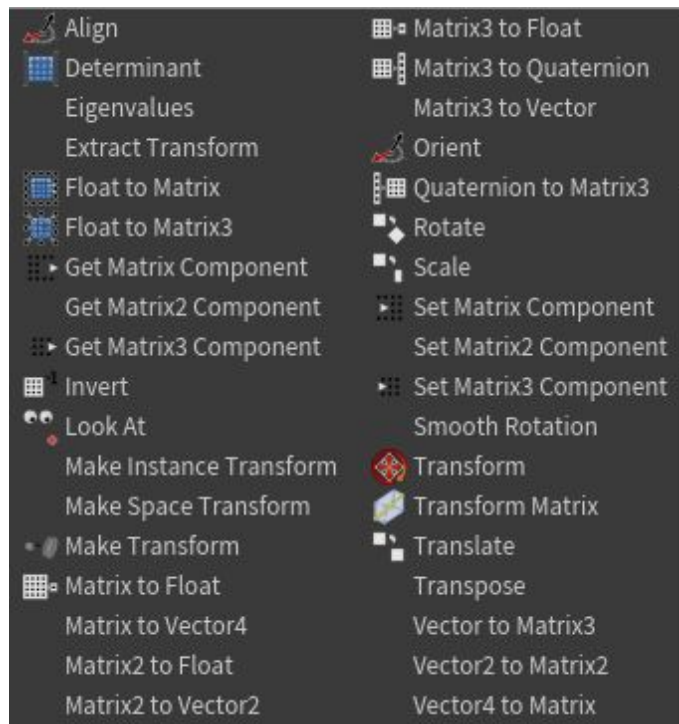
# Common Nodes

## Arithmetic/Aggregation



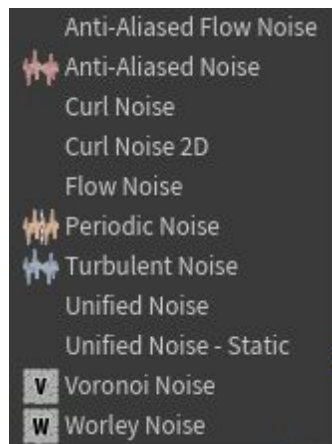
## Transform

Important nodes here are Look At, Translate, Scale, Rotate

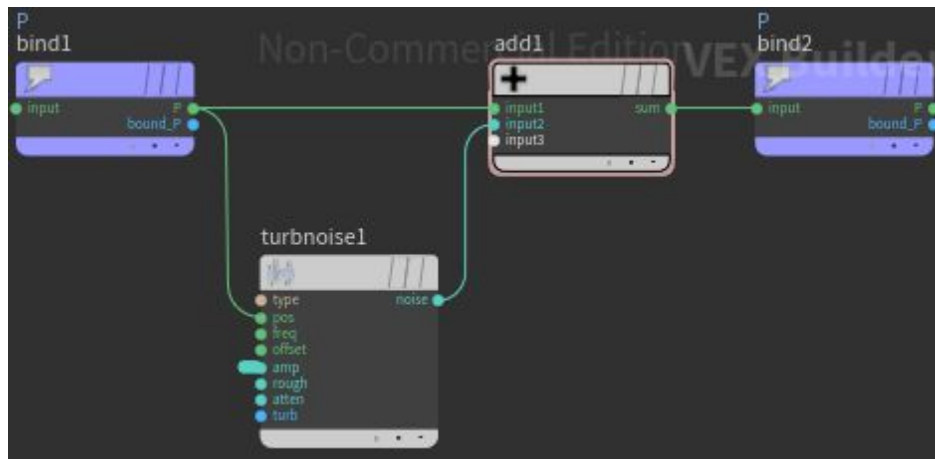


## Noise

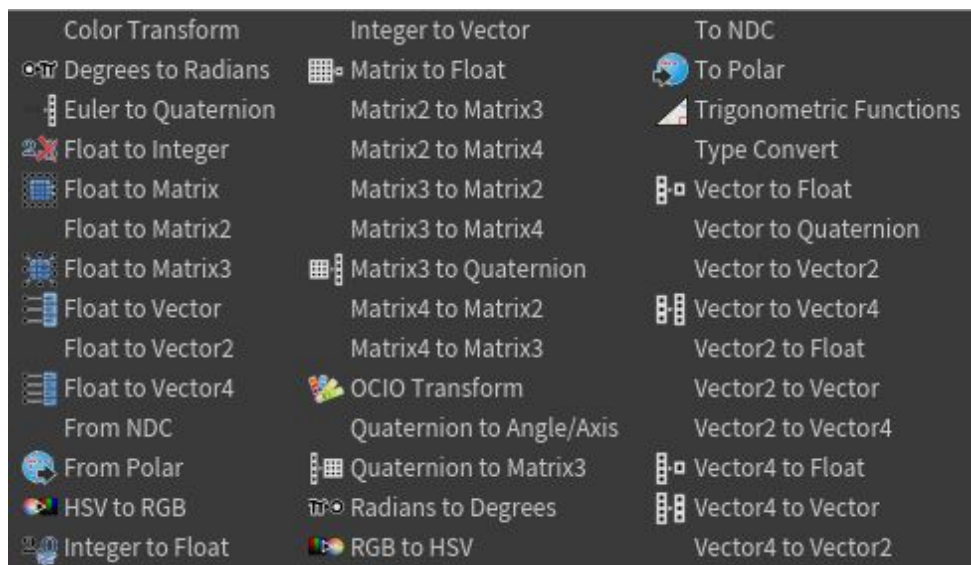
Import node here is Turbulent Noise.



Make sure you look at the documentation for the noise before using it. If you're feeding in positions, the output is the position offset -- you have to add the offset to the original value to get the final result...



























## Conversions



## Utility

Important nodes here are the ones used for comparison and fitting (shrinking down values to some range)...

- Comparison: Switch, Compare
- Fitting: Fit Range, Fit Range Unclamped

 Add Steer Force	 Fit Range	Ramp Filter
Add Wind Force	Fit Range Unclamped	 Ramp Parameter
 And	Gain	 Random
Anti-Aliased Ramp Parameter	Gaussian Random	Random Sobol
 Bake Exports	Gaussian Random UV	Report Error
Bias	 If Connected	 Return
 Bind	 Inline Code	Sample Sphere
 Bind Export	 Is Connected	 Snippet
 Box Clip	Limits	Soft Clip
 Clamp	 Luminance	 Struct
 Class Cast	 Method Call	 Struct Pack
 Color Map	 Method Subnetwork	 Struct Unpack
 Compare	 Mix	 Subnet Input Connector
 Constant	 Non-Deterministic Random	 Subnet Output Connector
Contour	 Not	 Subnetwork
 Copy	 Null	 Switch
CVEX Shader	 Or	 Two Way Switch
 Depth Map	 Output Variables and Parameters	Typed Two Way Switch
 Distance	 Parameter	Visualize
 Distance Point to Line	 Plane Clip	Wave Vector
Dual Anti-Aliased Ramp Parameter	 Print	 Xor
 Environment Map	Properties	



