

VEX

Introduced as a language to construct shaders.

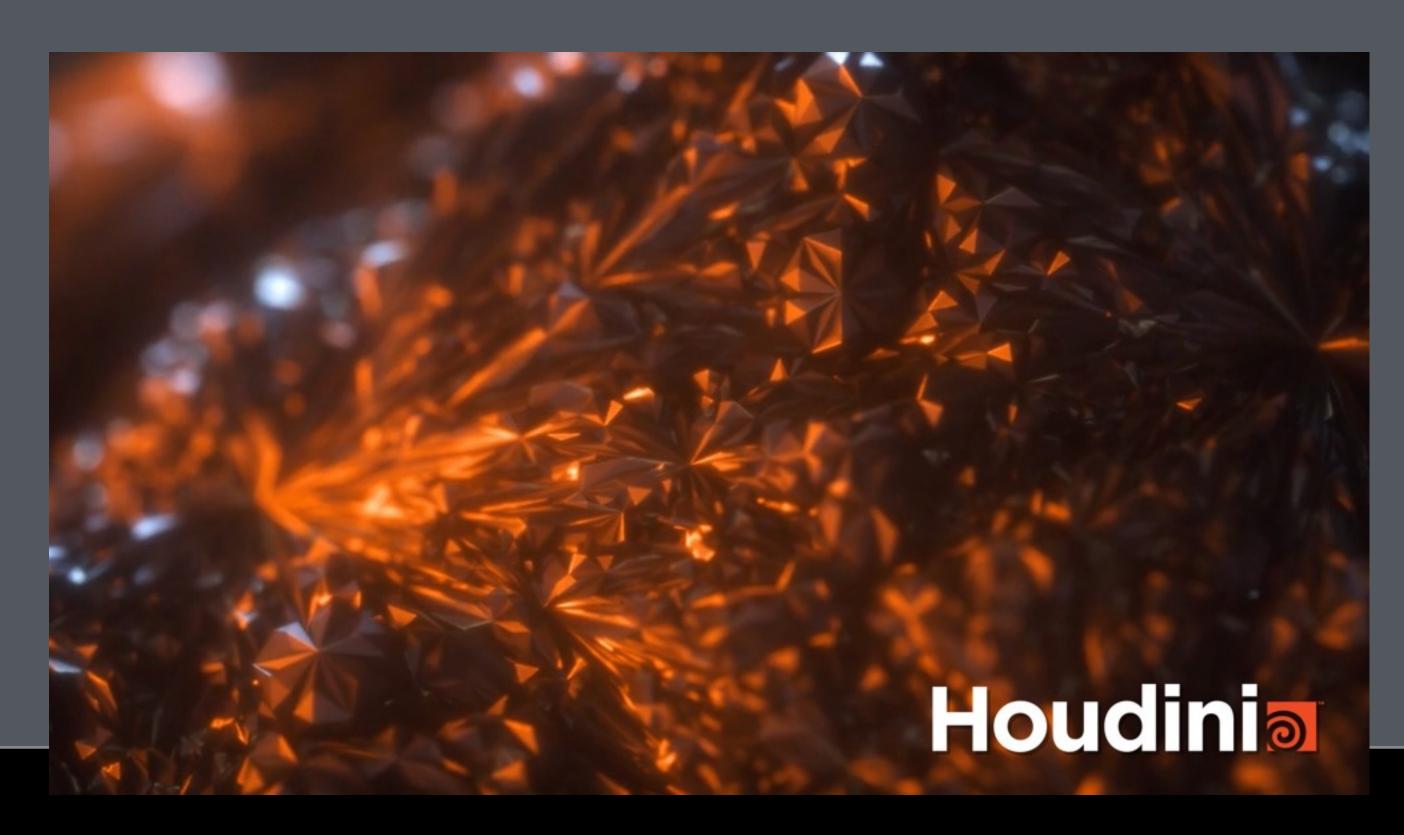
Expanded to SOPs to visualize shading functions on geometry.

Wrangling added to open VEX to everyone.

VEX is now everywhere!



vex snippets



Start Here:

www.sidefx.com/docs/houdini15.5/vex/snippets

Snippets = Wrangling = VEX Wrangling

VEX Wrangling means learning vex.

Snippet VOP enables wrangling parameter.



Useful Help Pages



www.sidefx.com/docs/houdini15.5/vex/cookbook www.sidefx.com/docs/houdini15.5/vex/lang www.sidefx.com/docs/houdini15.5/vex/statement www.sidefx.com/docs/houdini15.5/vex/arrays www.sidefx.com/docs/houdini15.5/vex/snippets www.sidefx.com/docs/houdini15.5/vex/geometry www.sidefx.com/docs/houdini15.5/vex/halfedges www.sidefx.com/docs/houdini15.5/vex/random www.sidefx.com/docs/houdini15.5/vex/strings



VEX WRANGLING



fetching input attributes



Fetching vector P

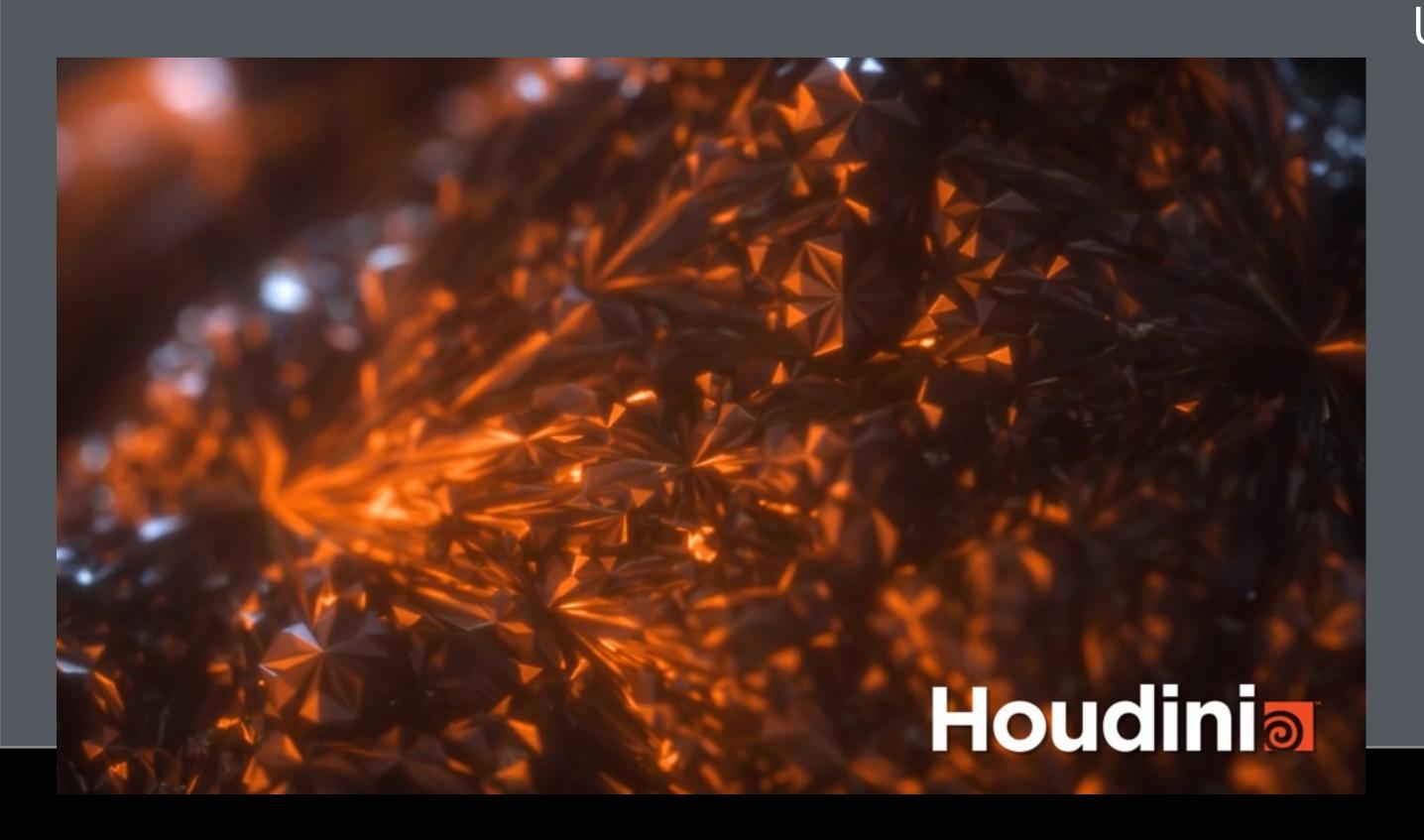
- @P // fetch first input P
- @opinput1_P // fetch second input
- @opinput?_P // fetch ?th input

Fetching float foo

- f@foo // fetch first input foo
- f@opinput1_foo // fetch second input
- f@opinput?_foo // fetch ?nt input



@opinput1_P vs point()



Use the point() vex function:

- to reference point attributes not in sync with the first input points.
- referencing points multiple times via common variables.
 - See Crowd Tool wrangling.



@ adding attributes



Exporting with @myattrib is always preferred where possible.

@myattrib is equivalent to the Bind Export VOP.

@myattrib streams and is fast.

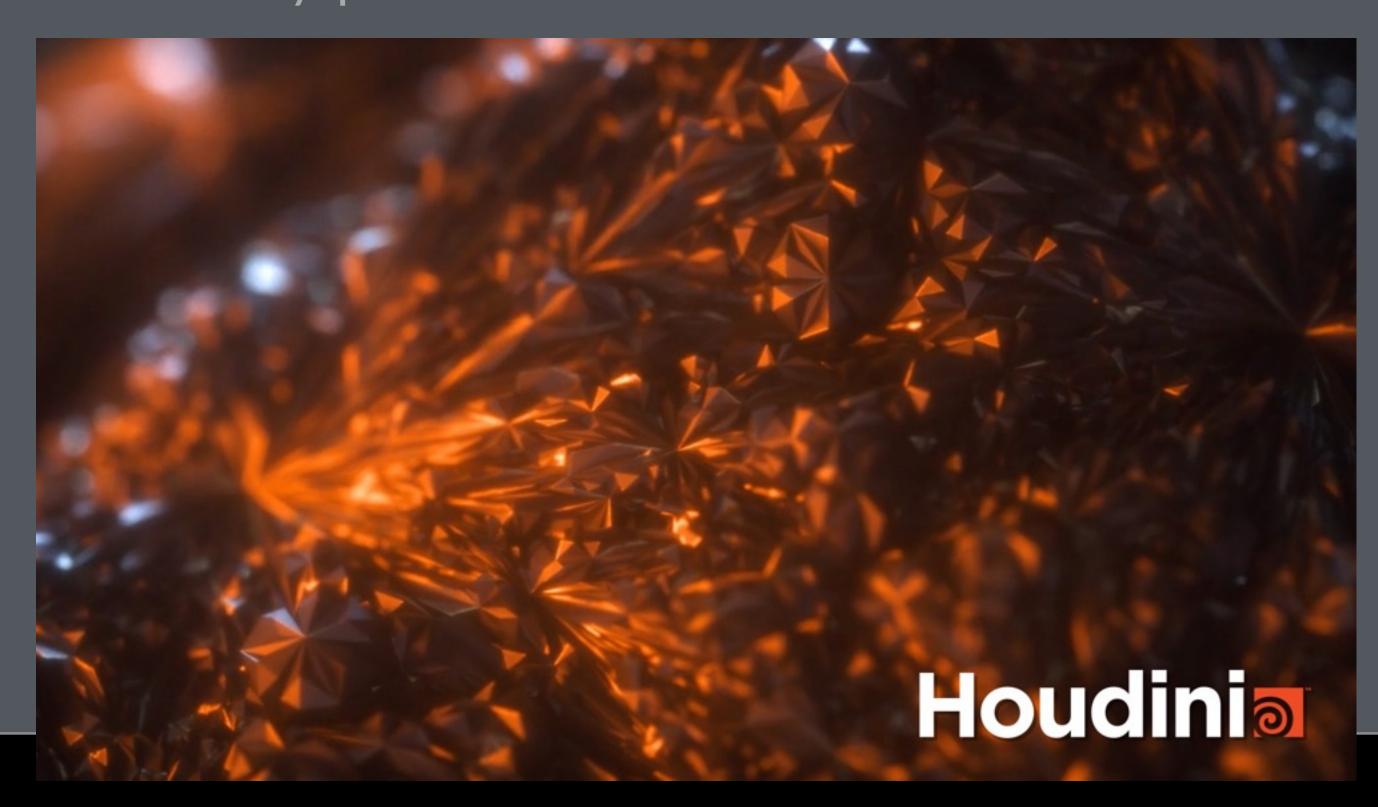
addattribute() vex function is slow.

Use setattribtypeinfo() for vectors, quaternions and matrices to set type for further downstream transforms.



VEX Webinar | Wrangling Attributes

setattribtypeinfo() attributes with types



Add attribute type info if you are dealing with vectors such as position, direction vectors, surface normals and quaternions.

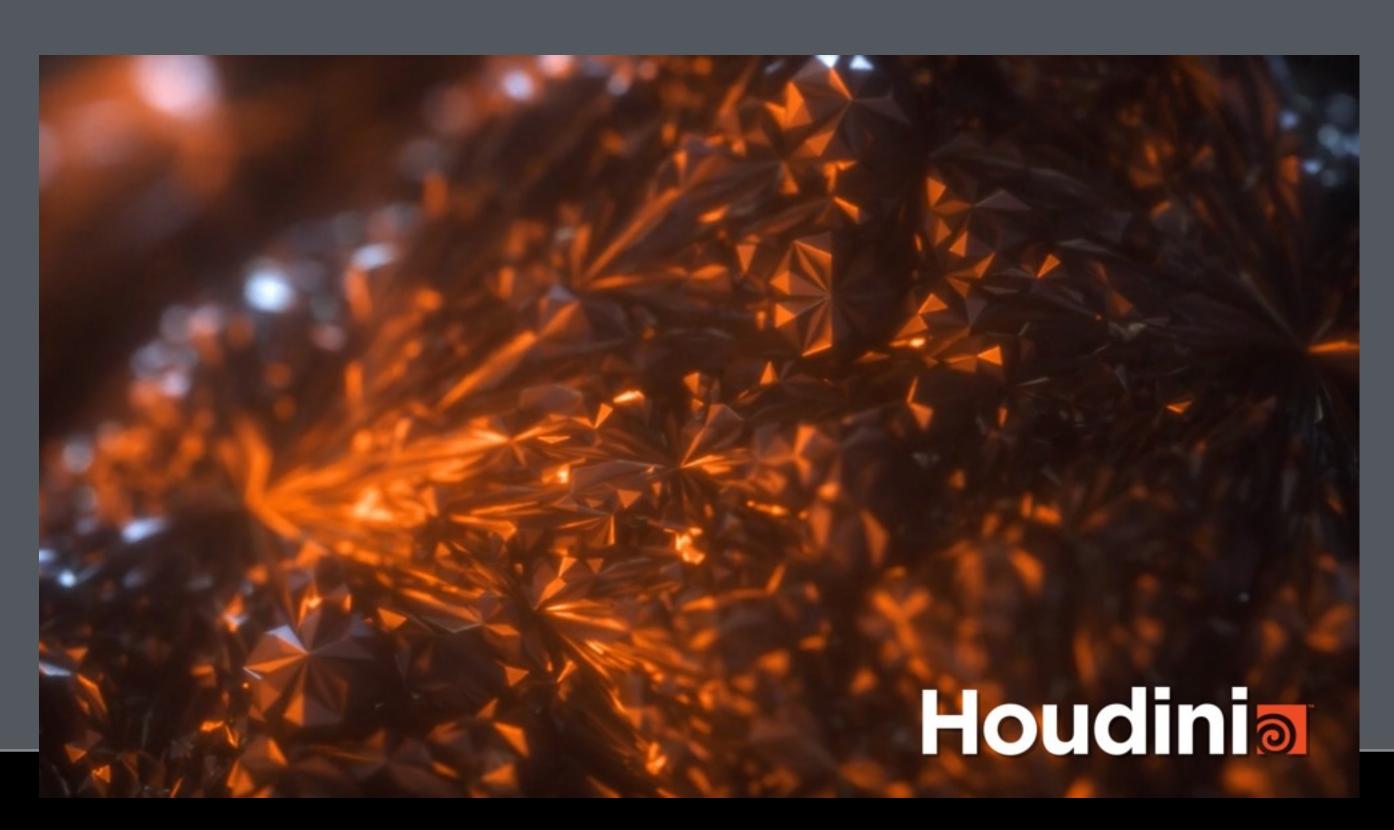
e.g.:

setattribtypeinfo(O, "point", "myattrib", point)

- Where O (zero) is geo handle to the input geometry or geoself(),
- "point" is the type of attribute from detail or global, point, point group, prim, prim group, vertex
- point is type info hints and can be one of none, point, hpoint, vector, normal, color, matrix, quaternion, indexpair, integer, integer-blend.



isbound attribute @ test



Inherit an attribute but if it is not present, set to a constant.

Same as Enforce Prototypes workflow.

e.g.: Input geometry may or may not have foo present. If it is present, inherit the values. If not, set to a constant.

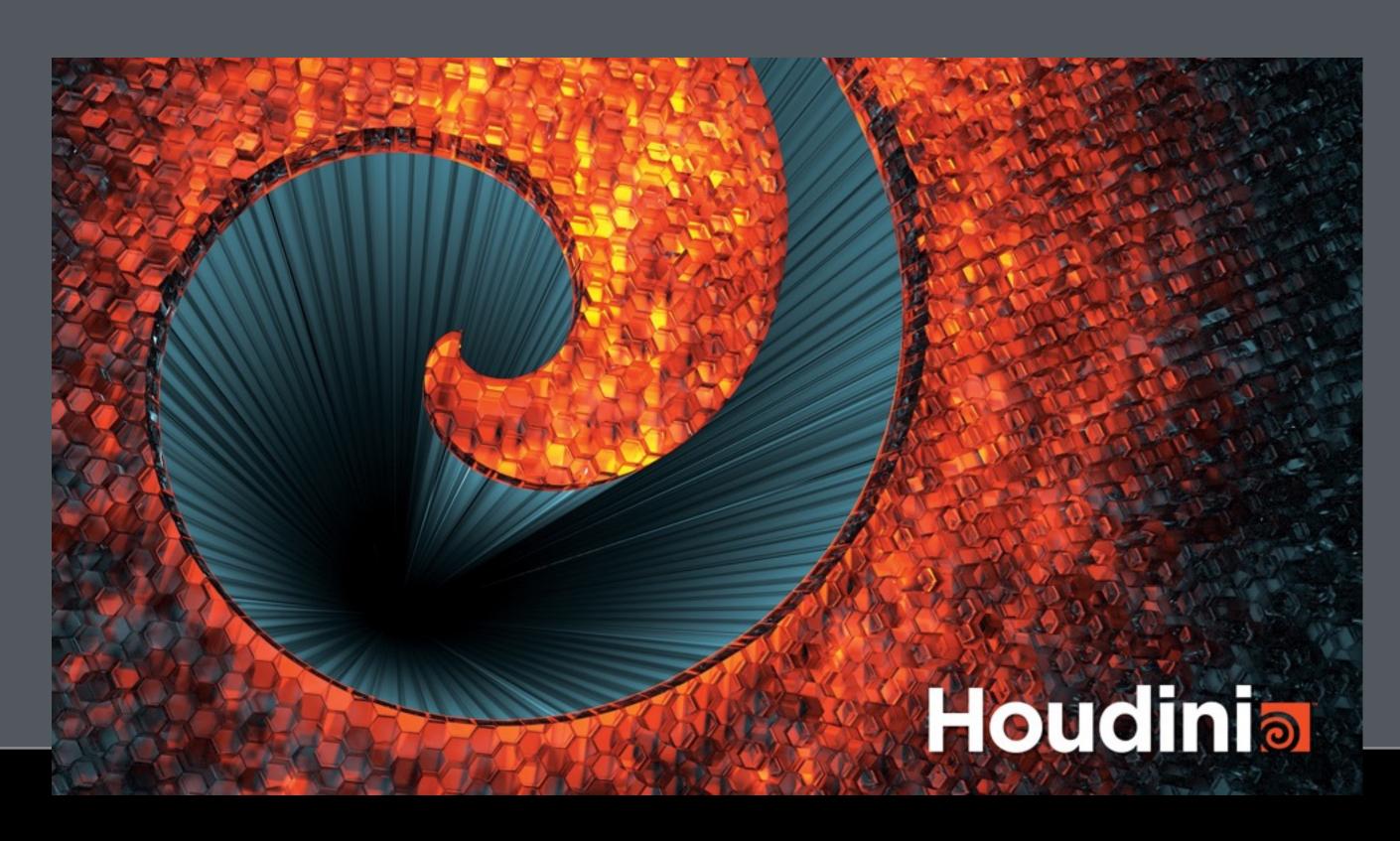
float @foo = 1.0; //inherit foo, set to 1.0 if missing

It must be set to a constant. Afterward you can modify as you wish.

Use isbound() for more complicated scenarios.



Groups and wrangling



You can fetch members of groups that match the given Run Over setting.

- run over points
- run over primitives

Use the @group_mygroup form to test group membership.

$$@Cd.x = (@group_mygroup==1) ? 1 : 0;$$

Use the @group_mygroup form to add and remove points or prims by O or 1 condition.



@group_ example



Examples of running over Groups



Functions using @group_



There are many functions that support groups with run_over set to the correct type:

- getbbox(), getpointbbox(), intersect(), intersect_all(), minpos(), pcfind(), relbbox(), relpointbbox(), xyzdist()

Use expandprimgroup() and expandpointgroup() to convert a group pattern into an array of primitive or point numbers.

pcfind() and nearpoint() support point group masks.



in-line vex functions

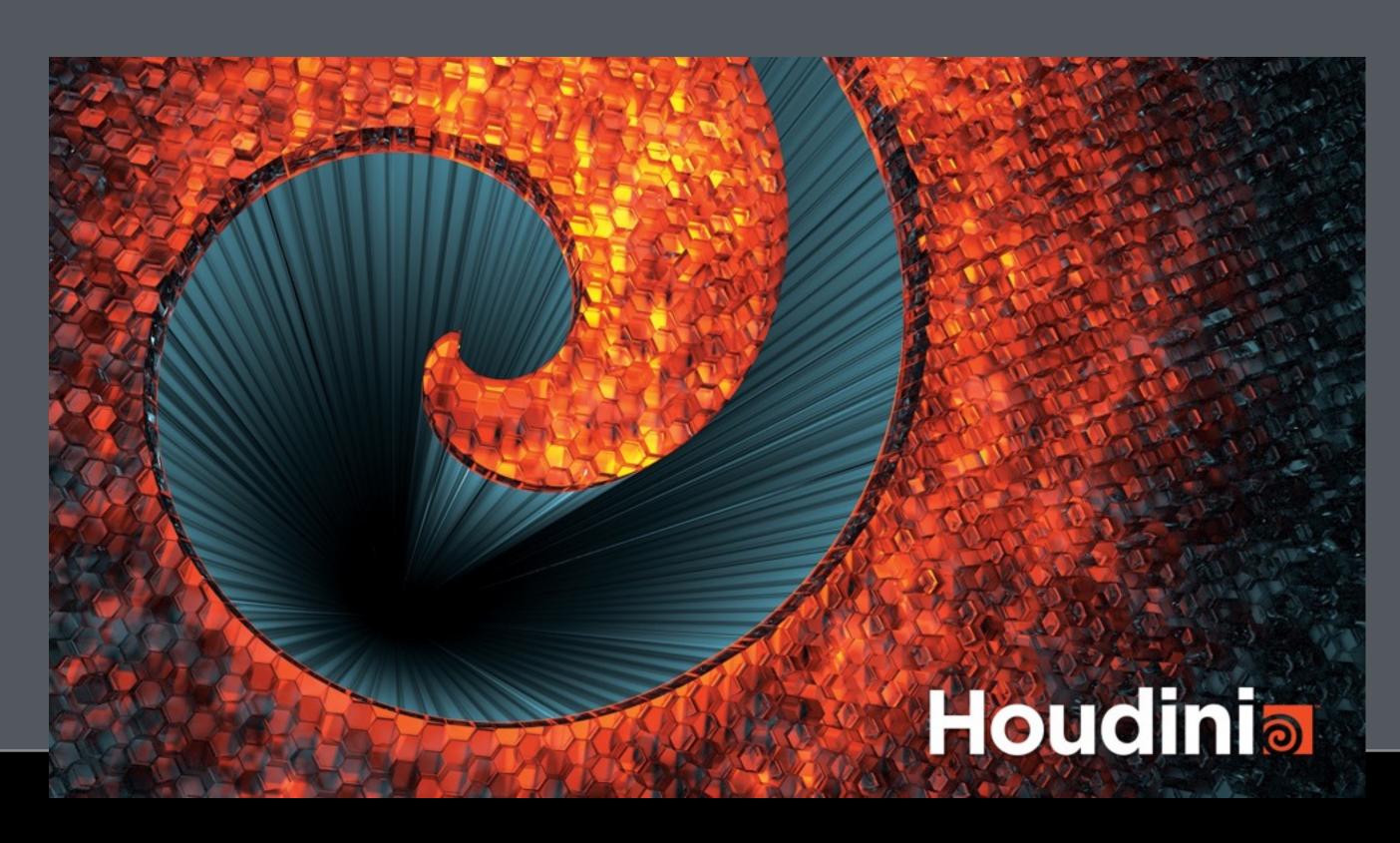


Define your own vex functions in Wranglers

```
e.g.:
float mysquare(float a)
{
   return a * a;
}
windresist = mysquare(windresist);
```



Trim your Export Parameters



Only use export @ parameters where needed.

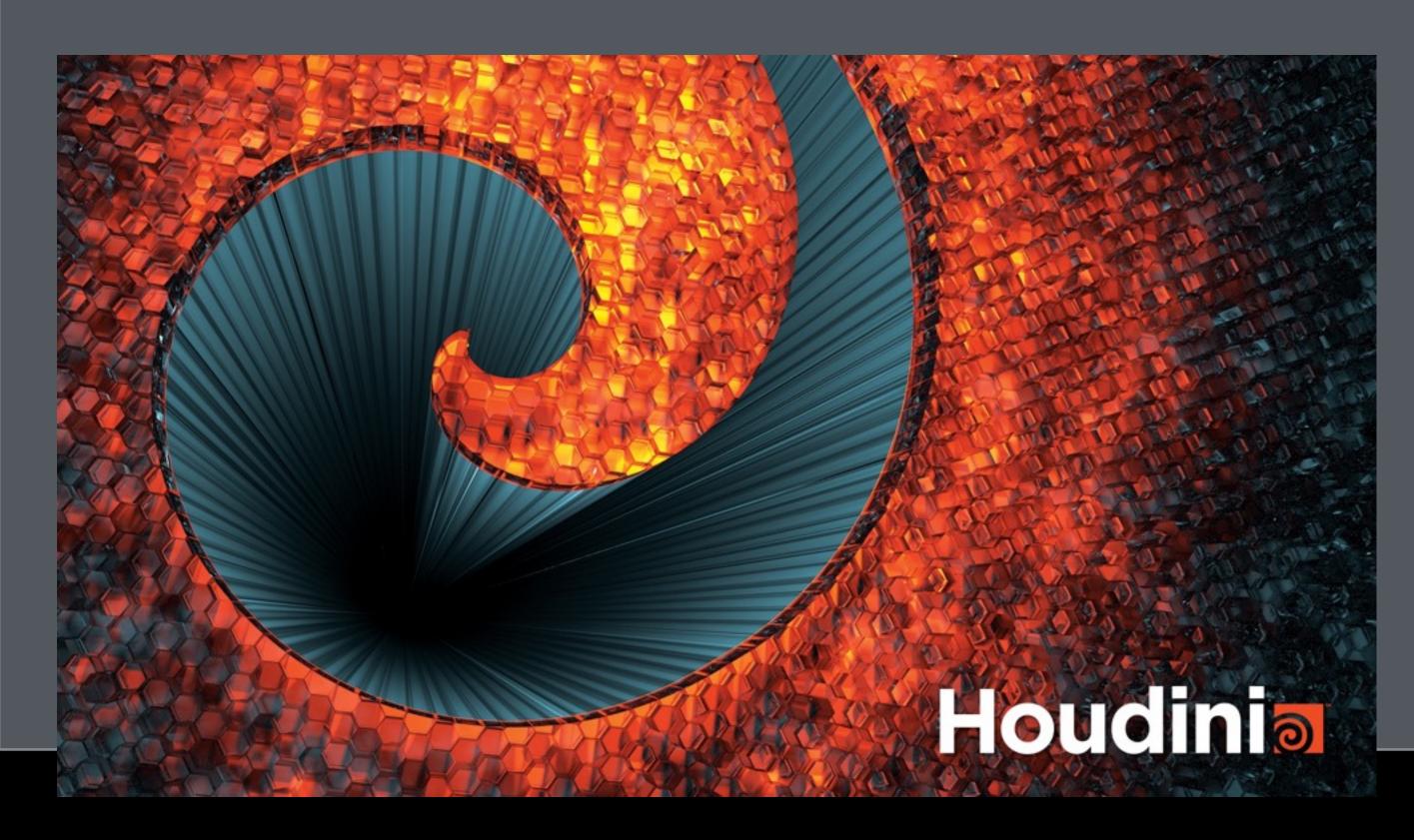
Don't write read only variables!

Every export attribute added to the geometry increases memory consumed.

e.g.: adding a vector type like Cd is the same as duplicating the geometry as P is a vector too!



export keyword



Use export keywords to force export of debug variables.



debug with sprintf()



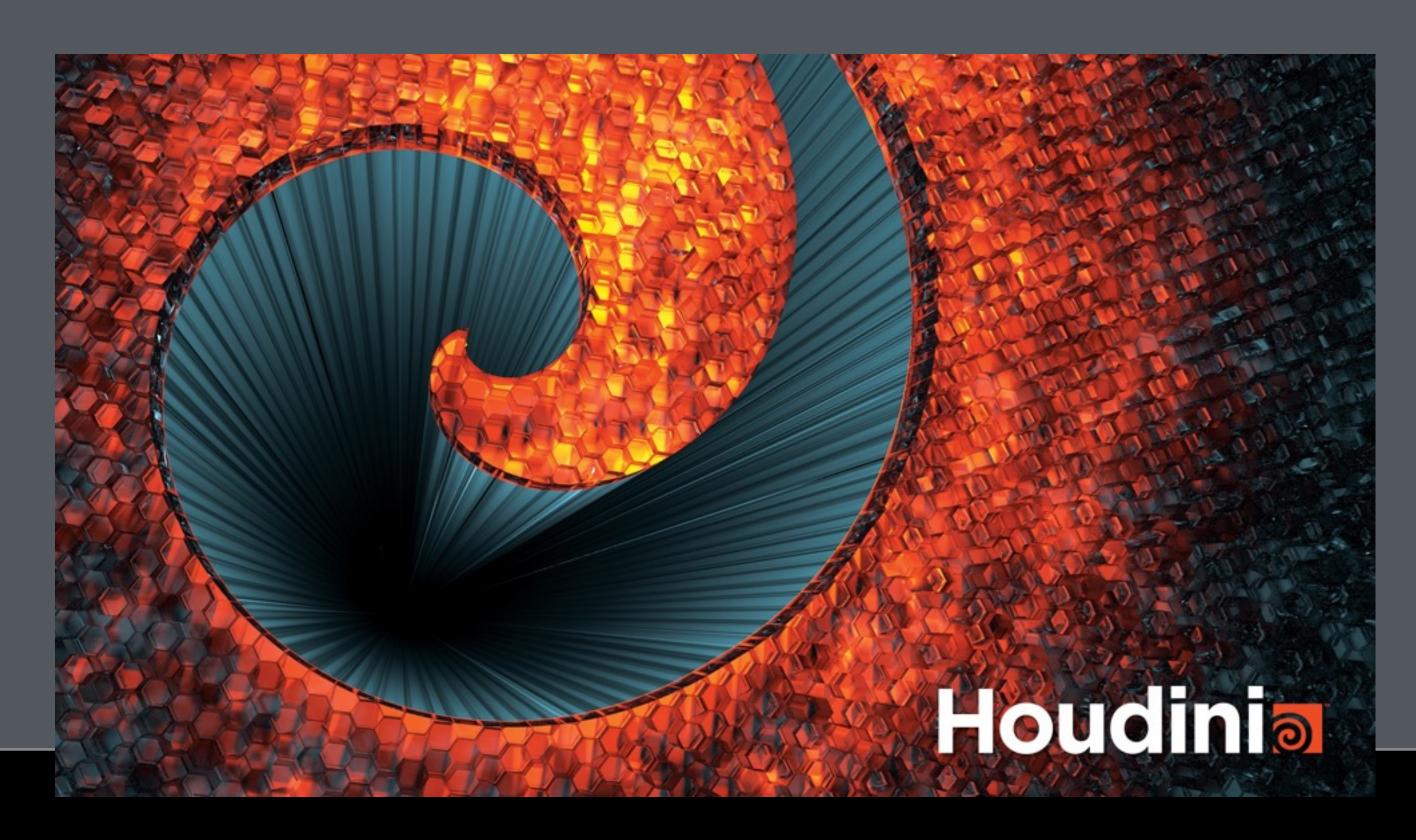
Debug vex code print out variables with:

s@debug = sprintf()

Do NOT debug with print()



error and warning functions

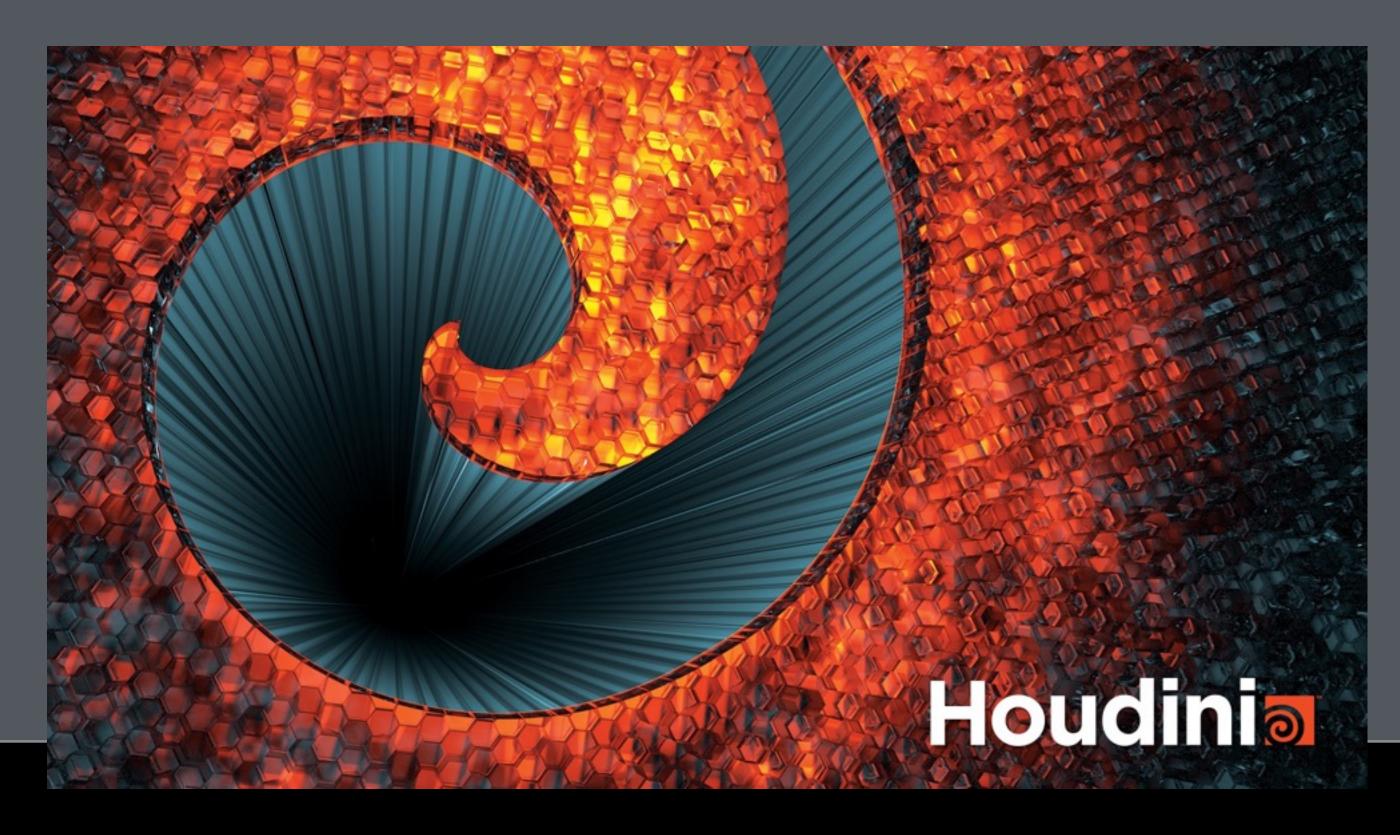


VEX functions "error" and "warning" to report custom errors and warnings at runtime in VEX.

Report Error VOP coming in H16.



Point Clouds and Arrays



pc wrangle friendly functions:

pcfind and pcfind_radius

pgfind()

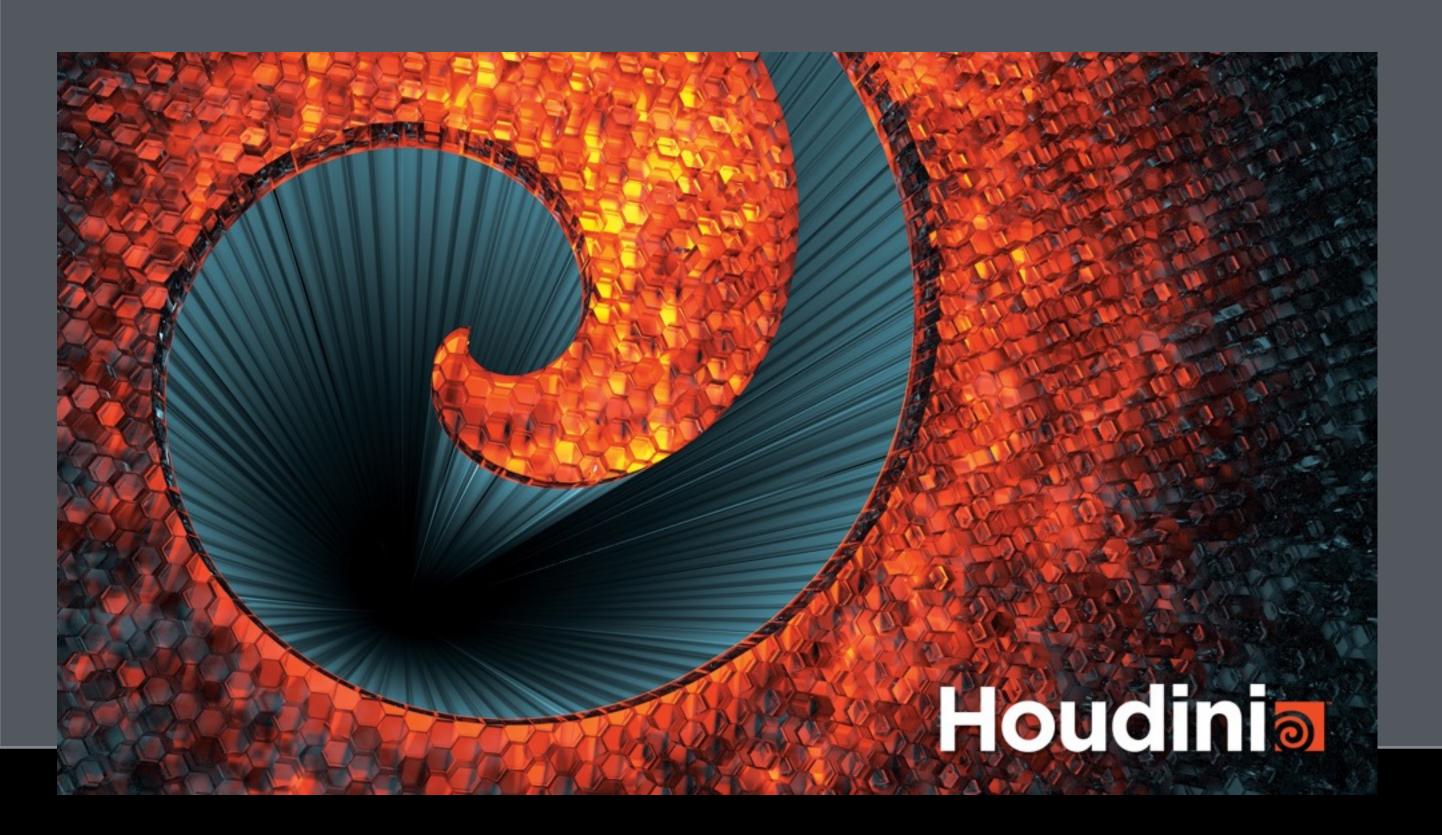
nearpoint()

nearpoints()

Iterating over pc Arrays



pgfind() for large uniform pc's

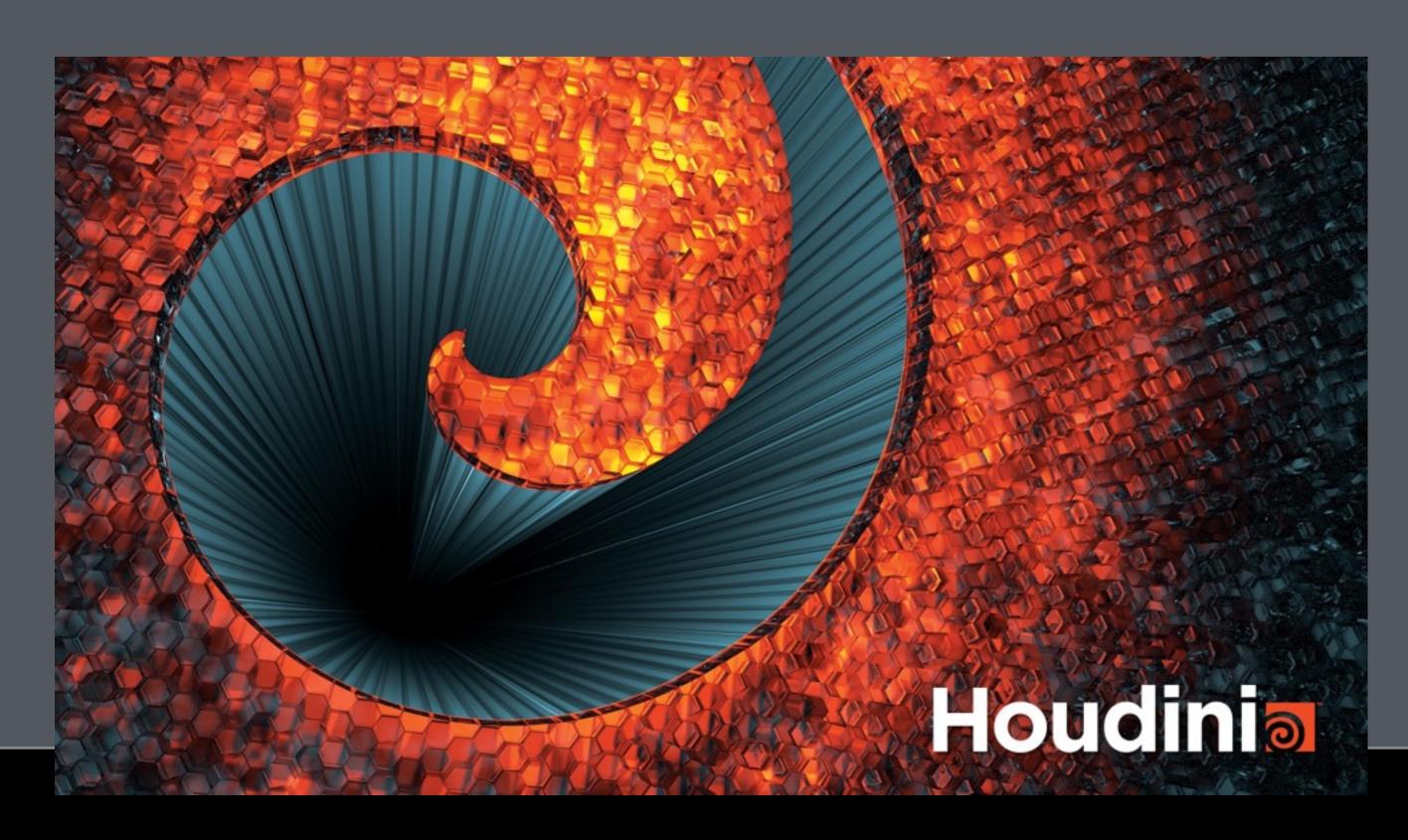


pgfind() efficient for massive point clouds where:

- > search radius is uniform.
- uses acceleration structure optimized for uniform point distributions.
- > set acceleration structure to search radius.
- working with millions of points in a pc.



pgfind() acceleration method



pgfind = Point Grid Find

This function buckets all points in to voxels of a specified size: the kernel radius.

To search, you just need to look in the voxels that overlap the search radius.

In Houdini nodes with "Assume uniform radius" on will switch to use pgfind() instead of pcfind().

H15.5 adds better support for larger search radii.



pgfind() uses



Only for optimizing a narrow use case for massive pc's and uniform radius look-up.

FLIP point data is uniform and great for processing with pgfind().

Massive wet maps where uniform radius is ensured.

Painting textures with massive pc's.



pgfind() issues



pgfind() is not efficient at all with varying radius look-ups.

pgfind() voxel acceleration structure overhead not applicable to pc's less than 1M points.

Tip:

 Continue using pcfind() and pcfind_radius() and only use pgfind() for performance on large uniform radius searches over millions of evenly distributed point clouds.



Wrangling Arrays



See Help page:

www.sidefx.com/docs/houdini15.5/vex/arrays



WRANGLE SOP FEATURES



Enforce Prototypes option



Enable when writing non-trivial wrangle code.

Protects you from typos with @ export attributes.

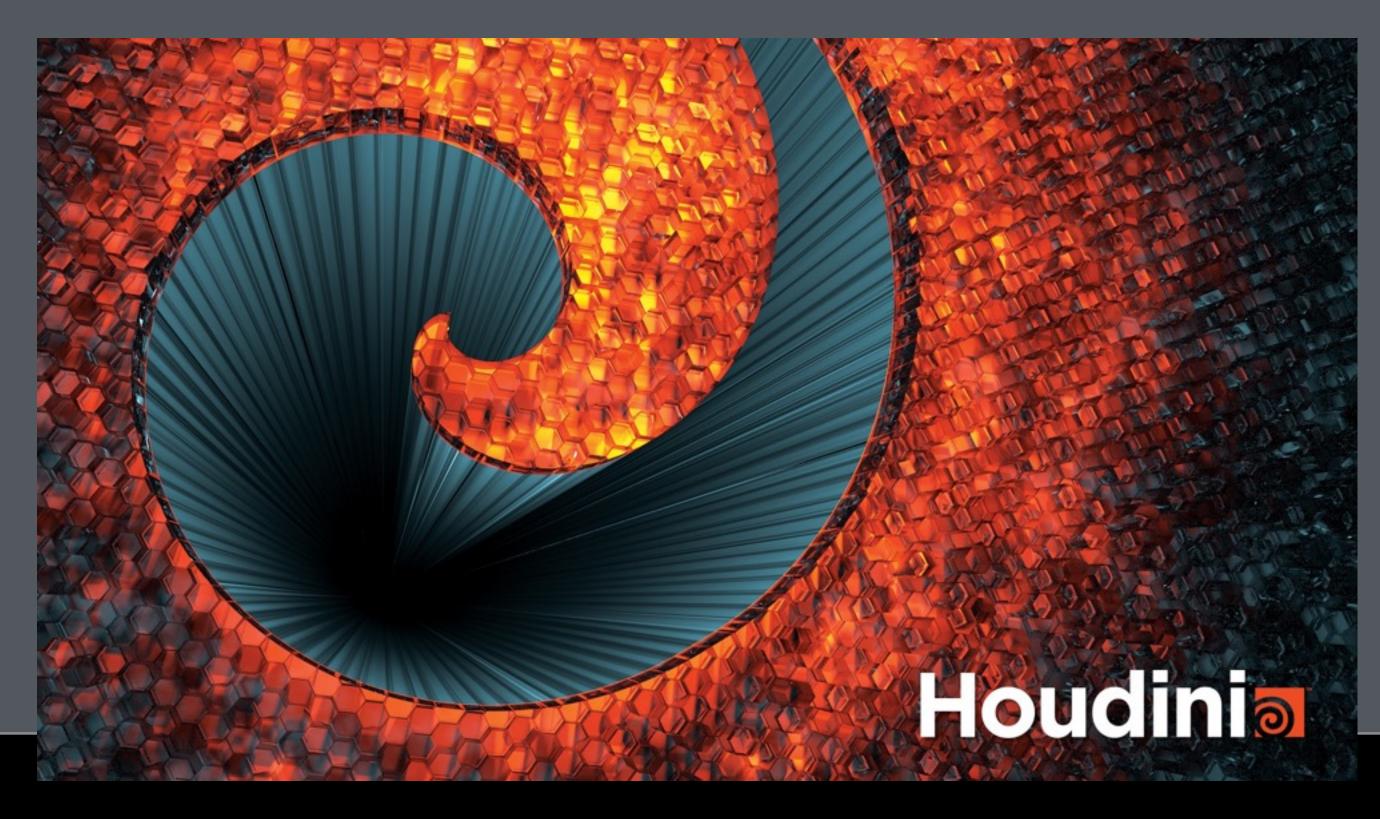
Subsequent use of @ vars without type prefix.

Useful for arrays like int[] where the prefix can look ugly.

All @ references must have a "prototype".



Enforce Prototypes option



Example with Enforce Prototypes enabled:

```
// declare ALL @ variables as prototypes
float @pattern;
int @ptnum;
vector @P;
vector @Cd

// code
float myrand = rand(@ptnum);
@pattern = noise(@P.x);
. . .
@Cd = @pattern;
```



Run Over Options



Run Over Points

Run Over Vertices

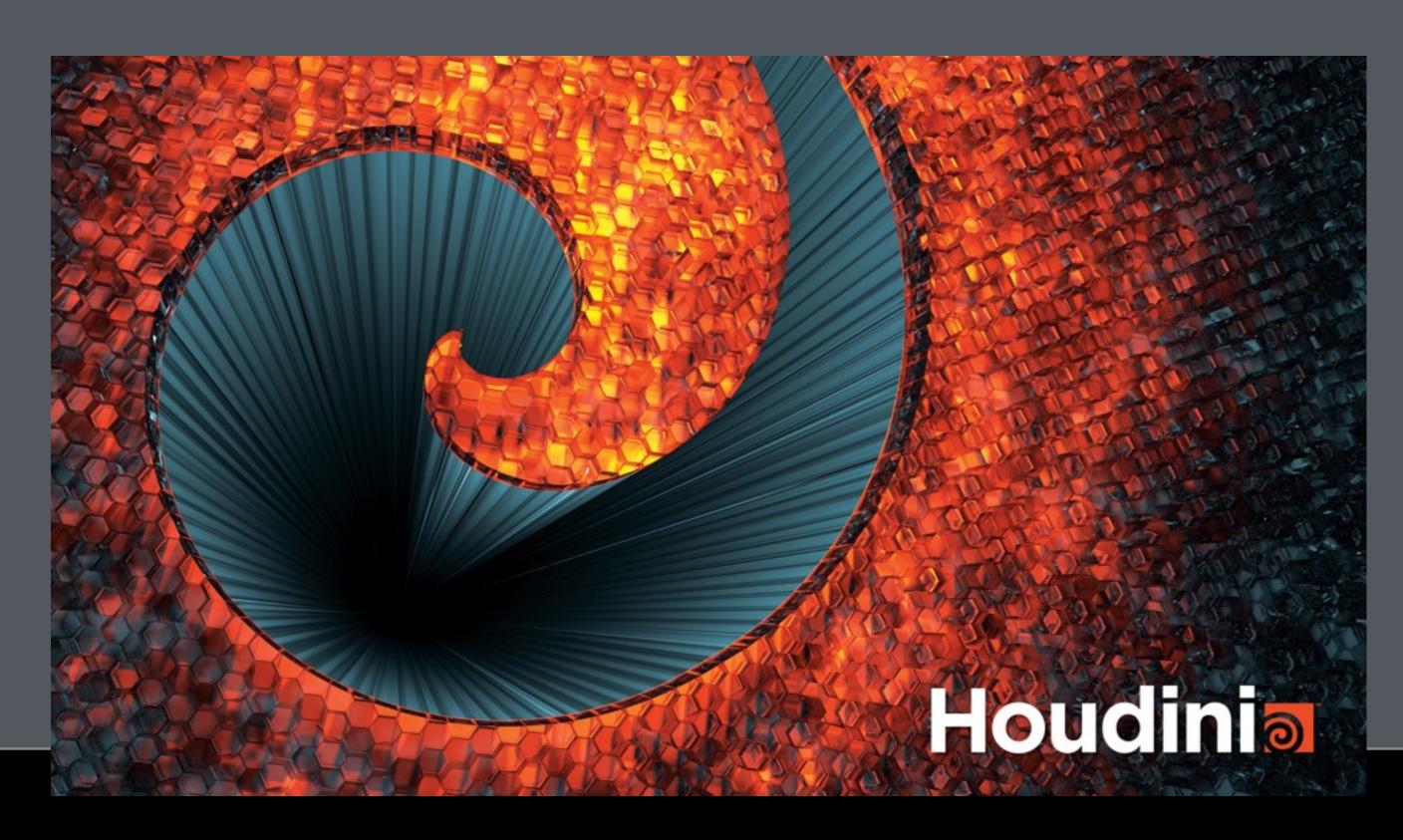
Run Over Primitives

Run Over Detail

- Only evaluates once
- lightweight



Run Over Numbers



When processing less than 1024

- for jobs that take a lot of time per point
- over a handful of points > crowd agents

Thread pool parameter

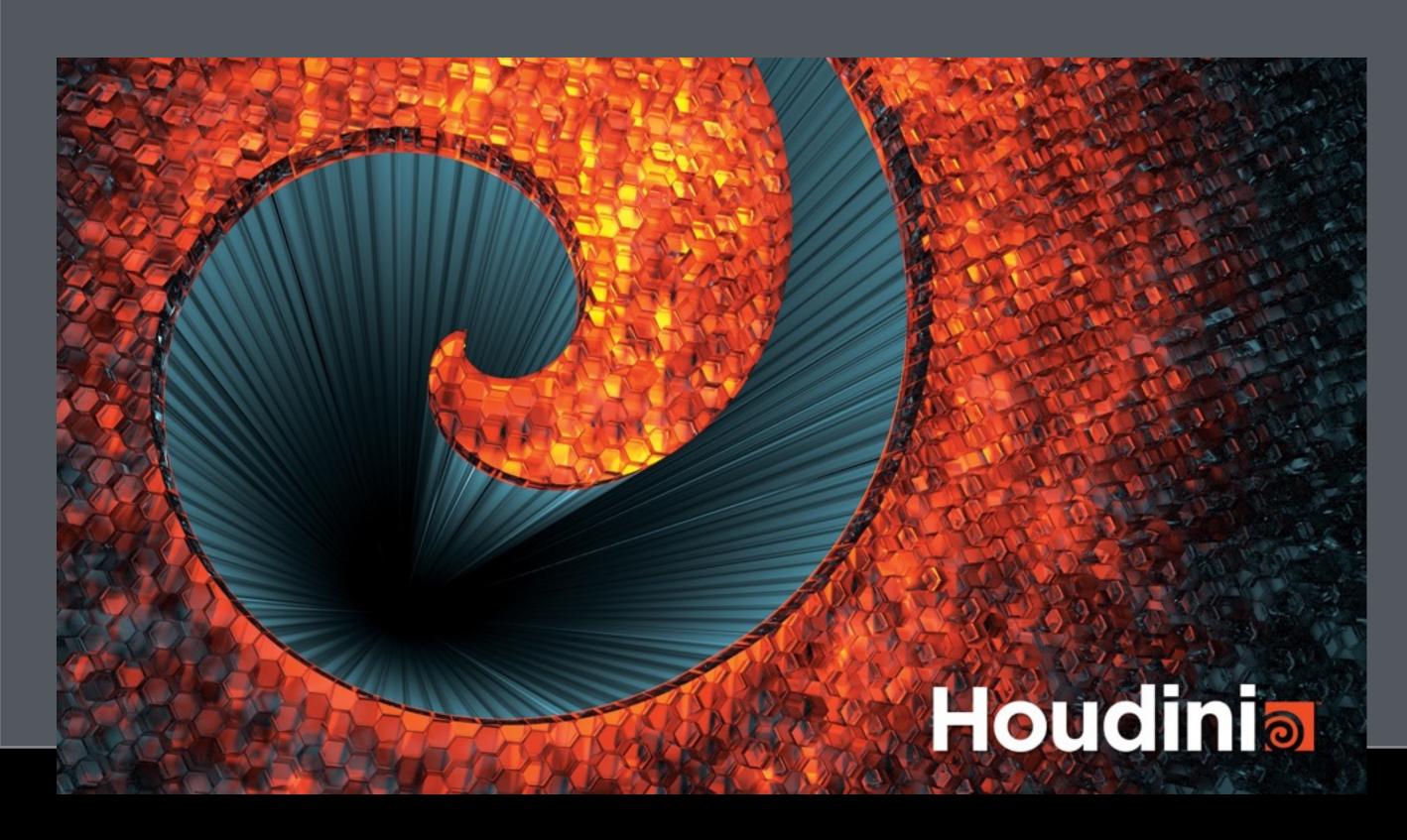
Specify number and it behaves like a detail, but your elem num binding returns current number.

Can thread Run Over Numbers

- thread non-point vex code.
- allows small thread pool sizes that can be split finely



Bindings Tab



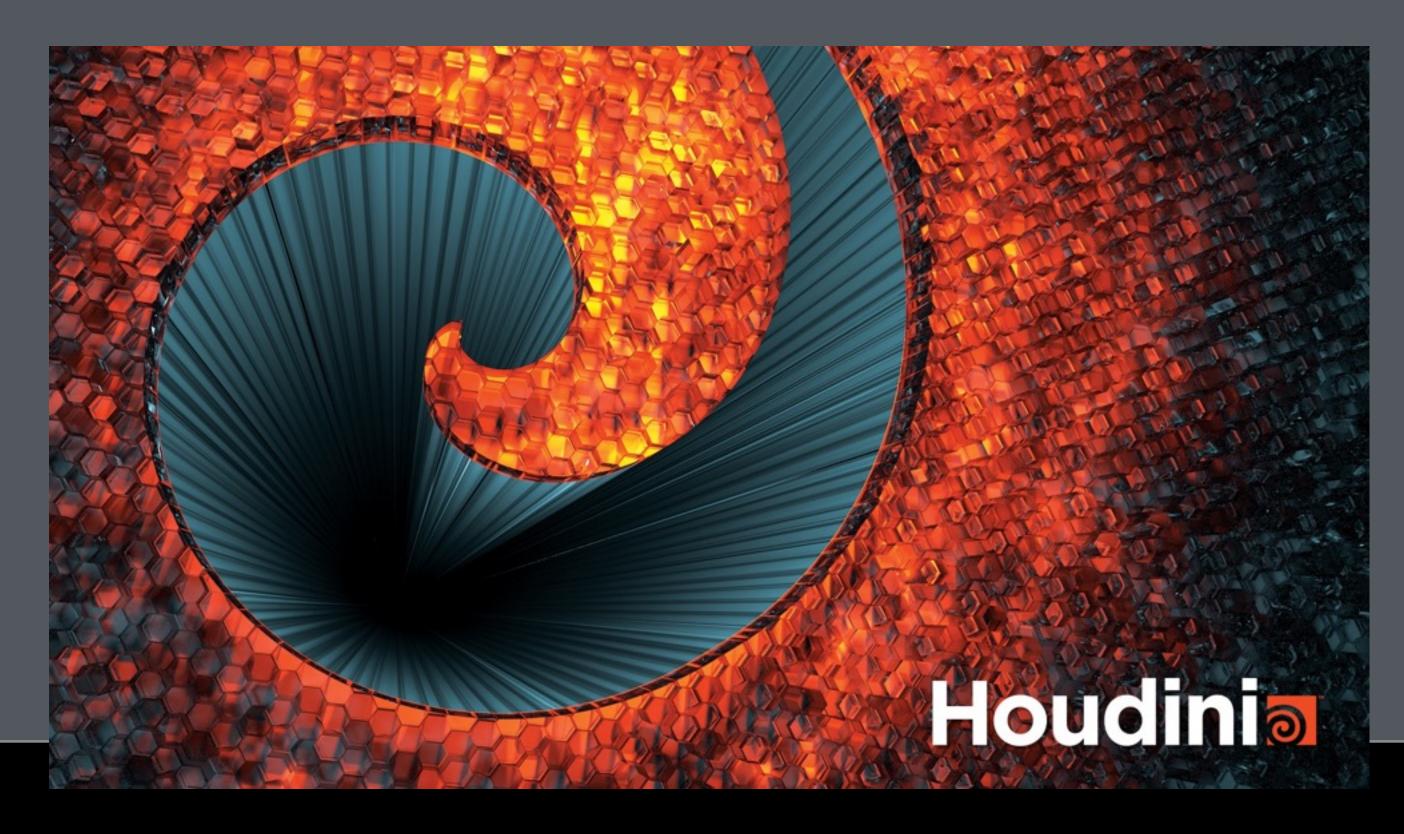
Used to turn your wrangle in to a reusable tool.

Map user defined input attributes to internal defined attributes.

Perfect for wrapping Wranglers inside HDA's with input user defined attributes.



hscript_ vex functions



VEX functions with identical output to hscript:

hscript_rand(seed)

hscript_turb(vector pos, int turbulence)

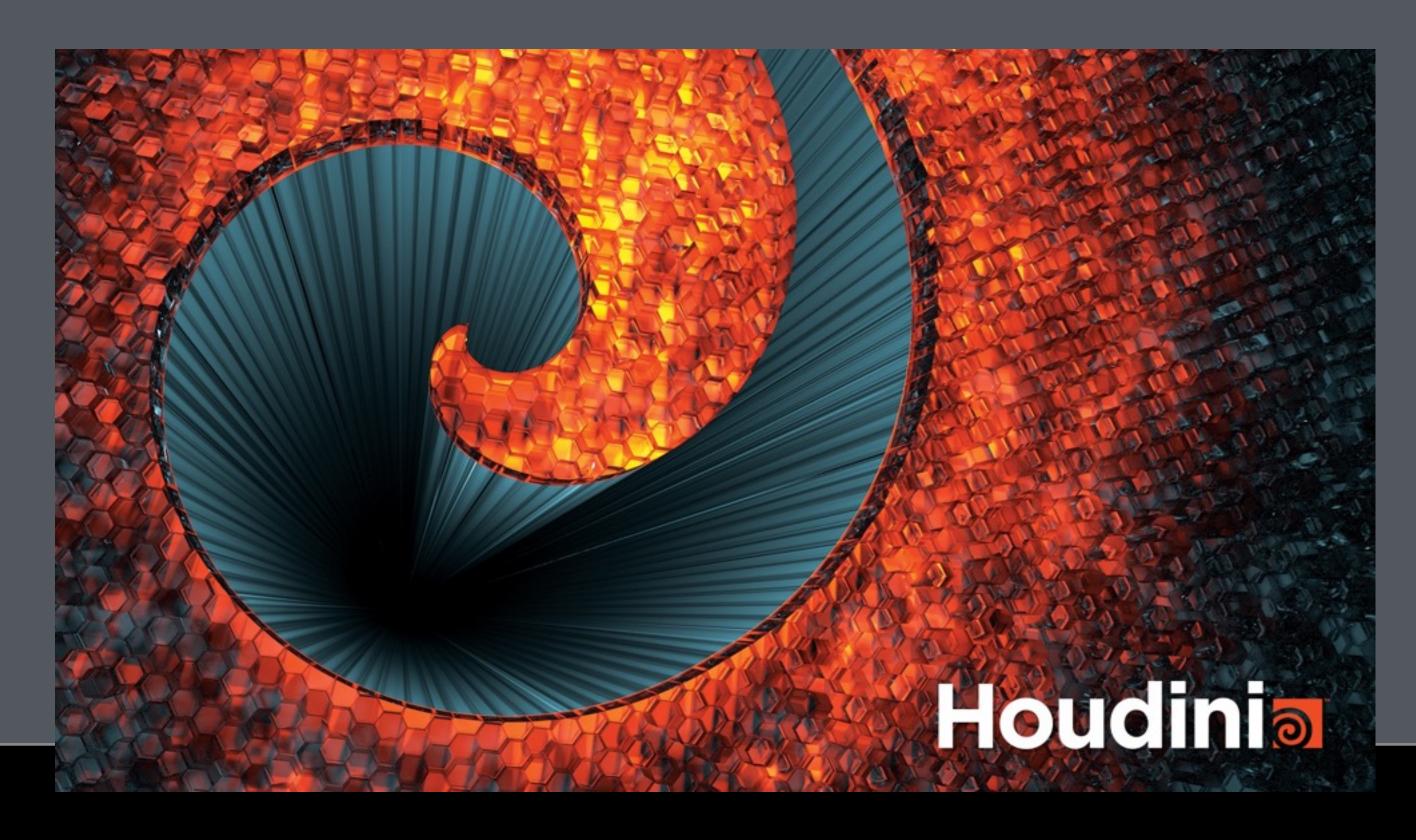
hscript_noise(vector pos)

hscript_sturb(vector pos, int turbulence)

hscript_snoise(vector pos)



VEX function list



List of VEX Functions internal to Houdini:
www.sidefx.com/docs/houdini15.5/vex/functions
Learn functions through application.



WRANGLE EXAMPLES



VEX Contexts

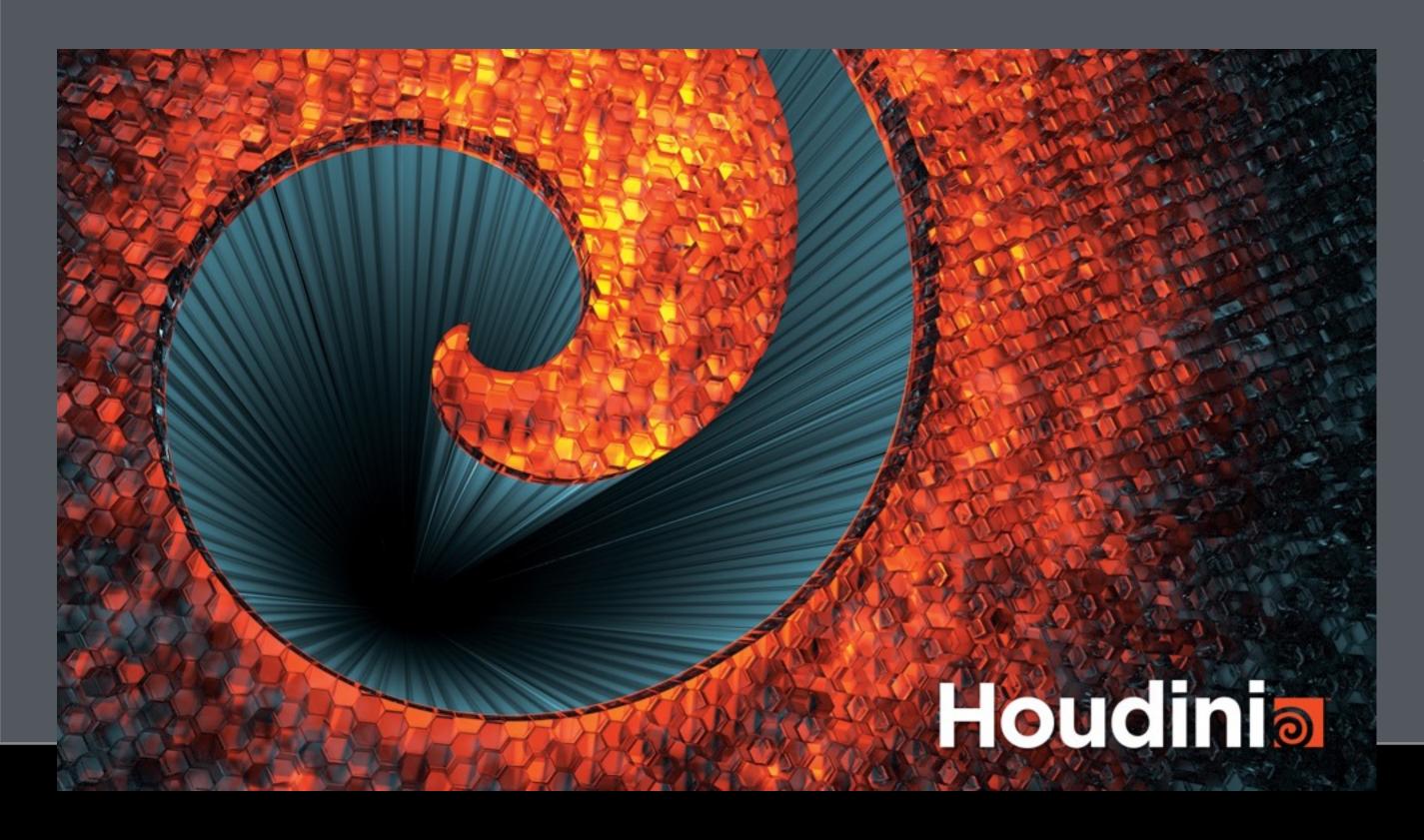


VEX is available throughout Houdini as a universal language to manipulate localized and generalized data (CVEX).

www.sidefx.com/docs/houdini15.5/vex/contexts



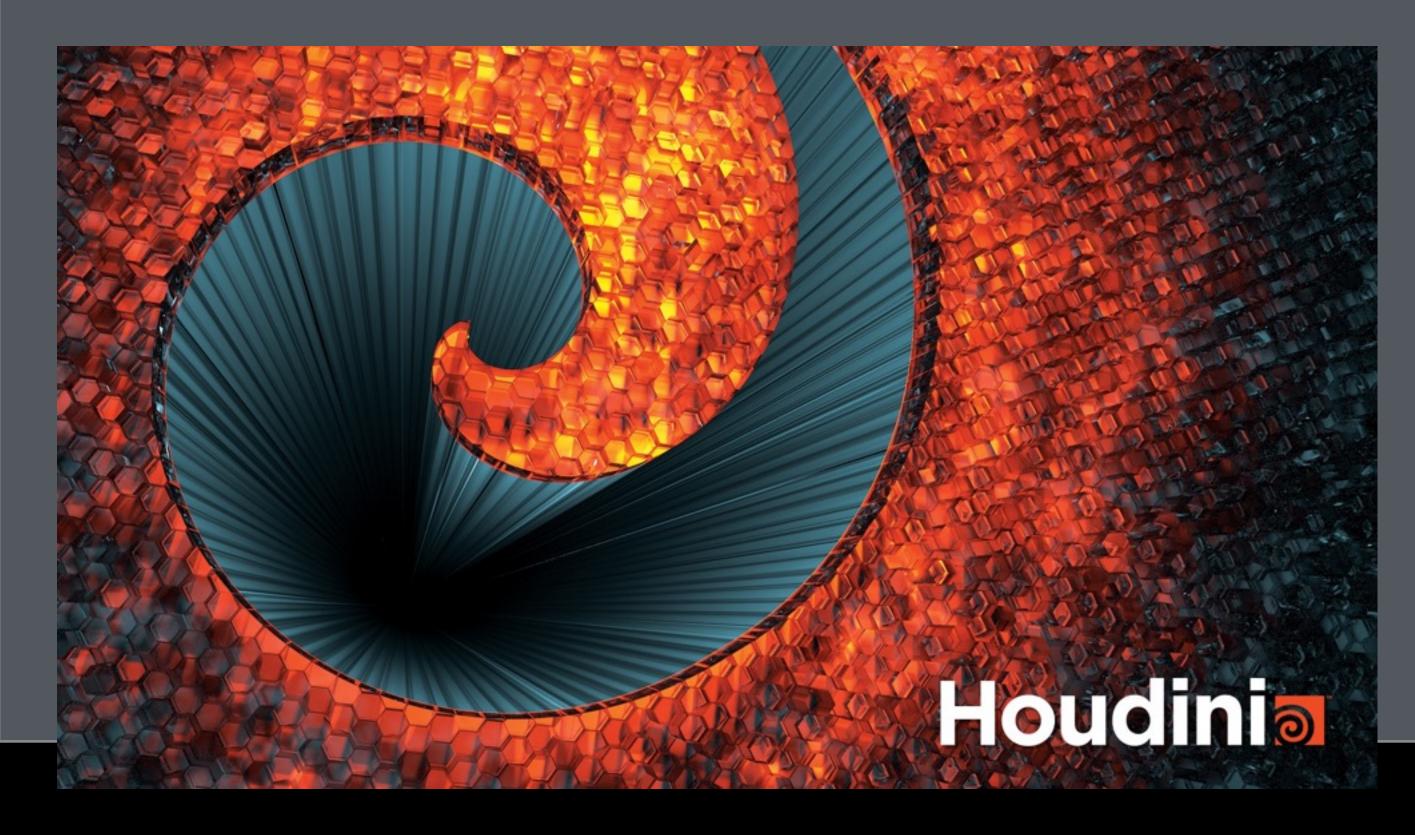
Style Sheet Wrangling



Wrangle VEX Expressions in Style Sheets



Future of VEX



Get a rand() that doesn't need an argument!

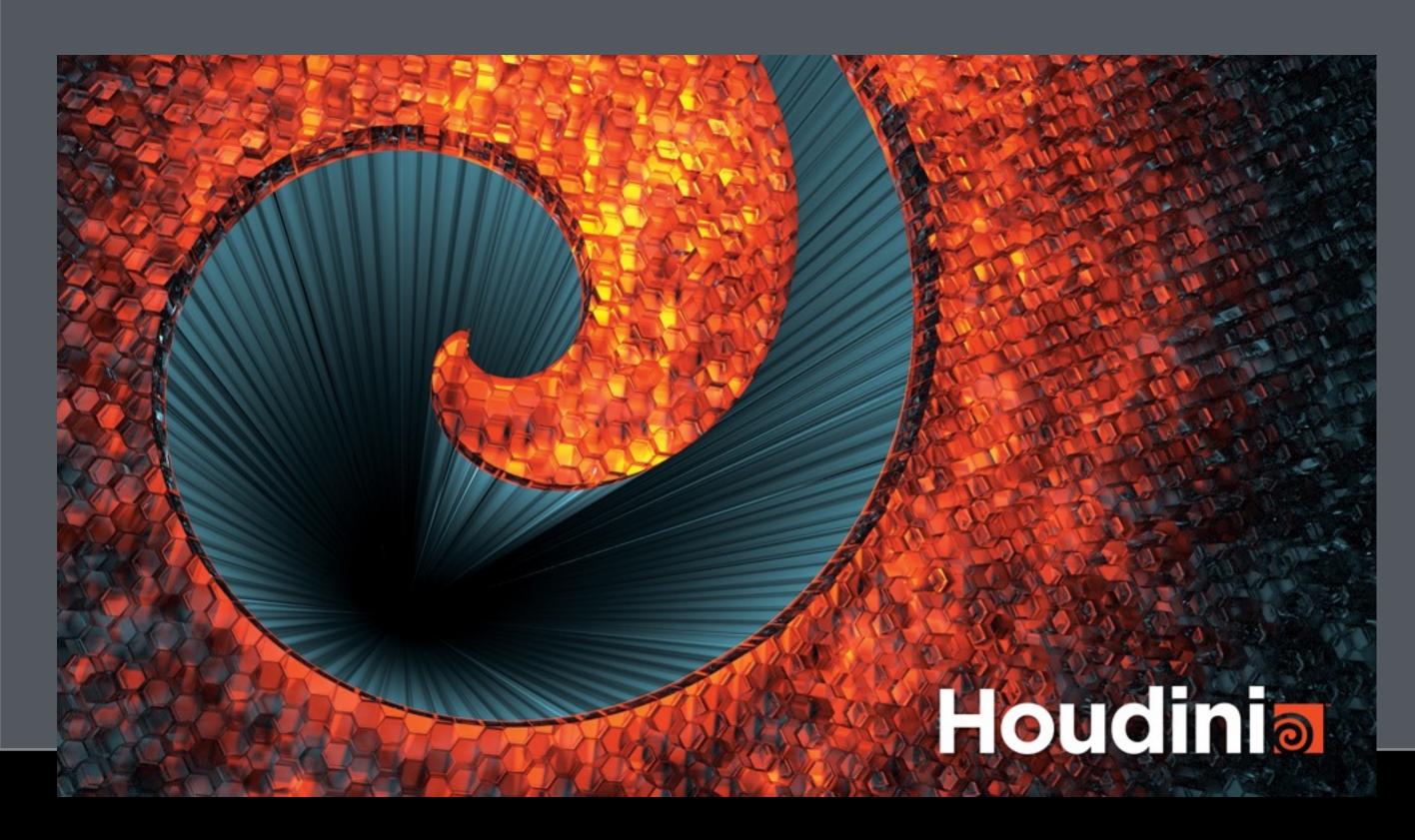
More geometry creators but no promises.



VEX TUTORIAL REFERENCE



VEX Learning Paths



Houdini Web Site Learning Paths

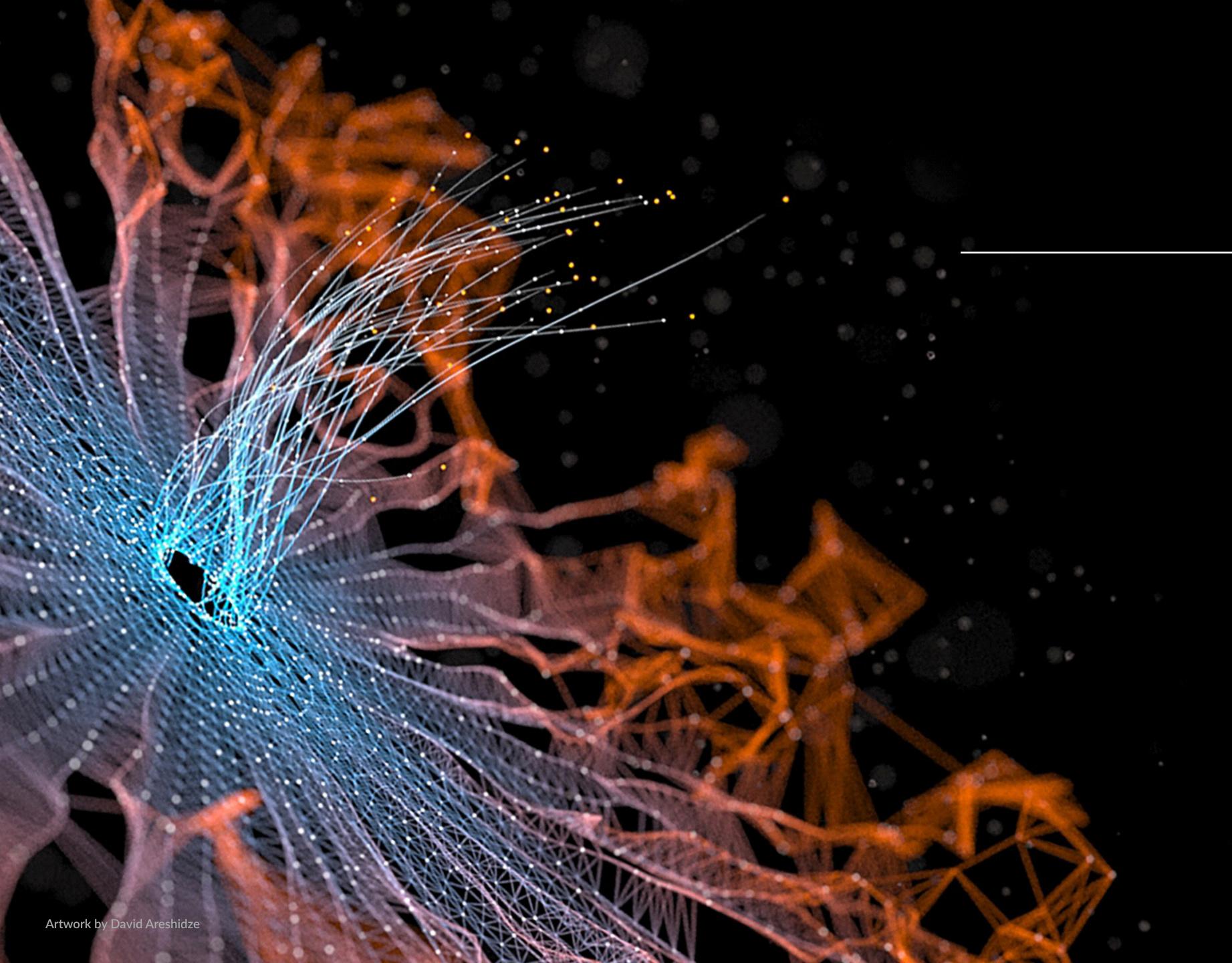
http://www.sidefx.com/learning-paths/

Vimeo > houdini vex

Entagma: Creating Geometry with VEX

https://vimeo.com/172529848





THANKYOU

Web: SideFX.com

Twitter: sidefx

Facebook: Houdini3D

