

# Houdini

## Expression functions

- 
- Houdini 



@ ~~TIME~~Snippets

Snippet

# Playback

\$	@	Description
\$FPS		Number of frames per second.
\$FSTART		Start frame number. <b>\$NFRAMES</b> must be greater than or equal to \$FEND - \$FSTART + 1.
\$FEND		End frame number.
\$F		Frame number.
\$FF	@Frame	Frame number.
\$NFRAMES		Number of frames to play. $\$NFRAMES = \$FEND - \$FSTART + 1$ .
\$RFSTART		Start frame number for reverse playback.
\$RFEND		End frame number for reverse playback.
\$T	@Time	Time in seconds. $(\$F - 1) / \$FPS$ .
\$TLENGTH		Total time in seconds.
\$TSTART		Start time in seconds.

\$	@	Description
\$TEND		□□□□□□□□□□

# General □□

\$ACTIVE TAKE		□□□□TAKE□□□□□□
\$E		□□□□ e (2.71828...).
\$HFS		Houdini□□□□□
\$HH		\$HFS/houdini.
\$HIP		□□□□□□□□□□□□
\$HIPFILE		□□□□□□□□□□□□□□□□
\$HIPNAME		□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
\$HOME		□□ Home □□
\$JOB		project directory.□□□□□□□□
\$PI		□□□□ pi (3.1415926...).

# Channels □□

\$OS		Operator String. Contains the current OP’s name.
------	--	--

\$CH	Current channel name.
\$IV	In value (value at start of segment).
\$OV	Out value.
\$IM	In slope
\$OM	Out slope
\$IA	In acceleration
\$OA	Out acceleration
\$LT	Local time - not including stretch or offset
\$IT	Start time of segment
\$OT	End time of segment
\$LIT	Local start time of segment
\$LOT	Local end time of segment
\$PREV_IT	Previous segment start time
\$NEXT_OT	Next segment end time

## COPs

\$CSTART	Start frame of the current COP.
----------	---------------------------------

<code>\$CEND</code>	End frame of the current COP.
<code>\$CFRAMES</code>	Number of frames for the current COP.
<code>\$CFRAMES_IN</code>	Number of frames available from the first input COP.
<code>\$CINC</code>	Gets the global frame increment value.
<code>\$W</code>	Current image width.
<code>\$H</code>	Current image height







## Render nodes

<code>\$N</code>	Current frame being rendered.
<code>\$NRENDER</code>	Number of frames being rendered.

# Houdini



Houdini  [VEX snippets](#)  [Python](#)  [VEX](#) 

HScript 	<a href="#">expression functions</a> 	
Python	<a href="#">Houdini Object Model</a>  	API  Python 
VEX		

 HScript  [Python Parameter](#)  [expressions](#).



-  
-  [Font node](#)  Variables are expanded 

`frame`padzero(5, $F)`.pic`

... `frame00001.pic`, `frame00002.pic`, 

 [expressions in filenames](#) 



 Houdini 

- `$F` (the current frame number) `$T` (the current time in seconds). [List of global variables](#)
- `HSc @pt @pscale` `pscale` (`pt`) `pt`
- `P` (position) `@P. x` `x/. y/. z` `1/. 2/. 3` `r/. g/. b`
- `@ptnum` `Point`
- `$HIP`

## 

HScript `ch`

[Spare parameters.](#)

...	
	<ol style="list-style-type: none"> <li>1. <b>Copy parameter.</b></li> <li>2. <b>Paste relative reference.</b></li> </ol> Houdini <code>ch(</code>
	<div> HScript <code>ch</code> </div> <div> <code>ch</code> </div> <div> <code>ch("tz")</code> </div> <div> <code>lamp</code> <code>X</code> </div> <div> <code>ch("/obj/lamp/tx")</code> </div> <div> <code>grid1</code> <code>Y</code> </div> <div> <code>ch("../grid1/ry")</code> </div> <div> <code>(chs.)</code> </div>
	<div> Name </div>



Script Wrangle Attribute Wrangle VEX snippet

VEX

Window ▶ HScript Textport

```
echo `expression`
```

Houdini Network “ ”

Print geometry node Y

Position Y @P.y + rand( @Frame \* @ptnum)

...

# HScript



( ) is not expanded. Text inside double quote " ) has variables expanded. A double-quoted string is considered one argument.

A backslash character ( \ ) escapes the next character. For example, to use double-quotes in a string:

```
"I had a \"great\" time."
```

When a string doesn't require variable expansion, use single quotes to speed up evaluation.



If you have two quoted strings next to each other with no spaces, they are considered a single argument. In this example...

```
set foo = "Hello world"
echo '$foo="'$foo'"
$foo=Hello world
```

...the echo command has one argument: '\$foo=Hello world'.

## Embedding

In the HScript command language, text inside backticks is evaluated as an expression. For example:

```
echo `strlen("$foo")`
```

### Tip

Scripting using the HScript command language is deprecated. You should use [Python](#) instead.

The string parser cannot decode nested quotes such as in the following (horribly contrived) example:

```
echo `system("echo `ls`")`
```

...however, it is possible to accomplish this with very careful usage of backquotes (and sometimes multiple backquotes in a row) to protect quote symbols from various levels of evaluation:

```
echo `system('echo `ls``')`
```