

# Houdini First Steps

## Module 05 - Procedural Animation

Ari Danesh  
ari@sidefx.com

SIDE EFFECTS  
SOFTWARE

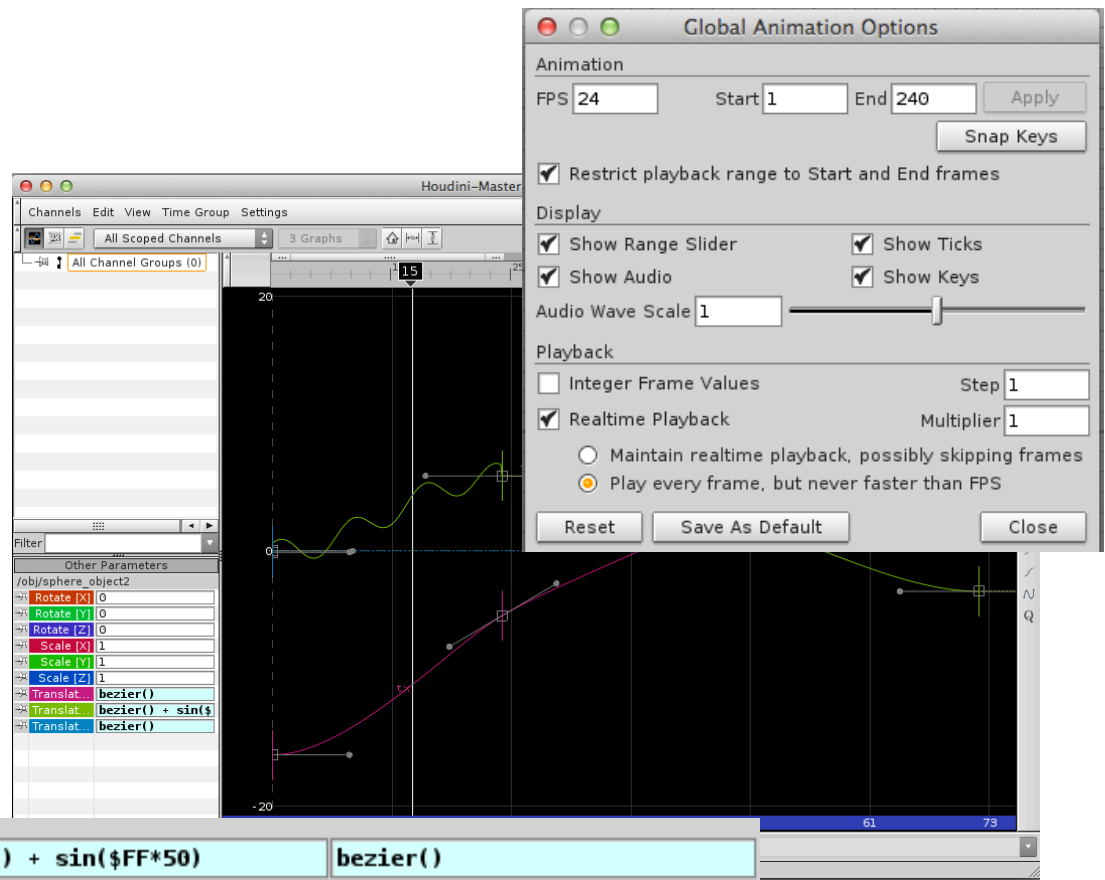


# Agenda

- **Adding Expressions to Keyframes (continuation from MO4)**
- **Data Types**
- **Variables**
- **Expressions**
  - exhelp
- **Projects**
  - Environment Project - Loading Sequential Files
  - Playbar Project - Display Timecode
  - Math Fun Project - Stretchy Hose
  - Standard Var Project - Unwrap Torus

# Adding Expressions to Keyframes

- For more accurate sim visualization -
  - In Global Animation Options turn off **Integer Frame Values** and make sure **Play every frame, but never faster than FPS** is turned on.



# Data Types

- int - 4, 18, 87 - whole numbers
- float - 32.124, 3.1416, 22.5 - numbers with a fractional component
- vector - (1,0,0) , (37.54, 14.83, 12.9) - a triplet of numbers

▸ vectors include

- Position
- Normals
- Color

Translate	<input type="text" value="3.00904"/>	<input type="text" value="2.0047"/>	<input type="text" value="-4.15252"/>
-----------	--------------------------------------	-------------------------------------	---------------------------------------

- String - "Hello World" , "We Love Houdini!" - in other words text
- Boolean - 1, 0 - Indicating True or False
- Matrix, a 4x4 array of numbers (advanced, beyond scope of course)

$$\begin{bmatrix} 5x & 3y & 4z & 2x+y \\ 3z & 32 & 12(3X+Y) & x^3 \\ 1 & 14 & 3s & 16 \\ xyz & 76y & 18z & 13xz \end{bmatrix}$$



# Variables

- Environment Variables - Variables for setting up the Houdini Environment
- Global Variables - Data that is accessible in all contexts. Standard Variables
- Playbar Variables - Various variables that are associated with time in Houdini
- Standard Variables - Commonly used variables that are unique to specific contexts, however are common enough that they may be recognized across multiple contexts
- Local Variables - Many variables that are common to one context are not necessarily common to another context. Not all variables are standard. In fact, some nodes have local variables, which will be listed in the help cards on a node-by-node basis

# Environment Variables

- ▶ Houdini uses environment variables for various purposes, such as defining paths (sets of directories in which Houdini looks for certain types of files), commonly used directories (such as \$HFS, the install location of Houdini) and for obscure or highly technical settings.
- ▶ Use the shell command - `hconfig` to see the variable settings
- ▶ `~/Library/Preferences/houdini/X.X/houdini.env`

```
Resources — Houdini Shell — bash — 115x28
Aris-MacBook-Pro:Resources aridanesh$ hconfig
EDITOR                               := '/usr/local/bin/bbedit'
HFS                                  := '/Library/Frameworks/Houdini.framework/Versions/12.0.599/Resources'
HOME                                := '/Users/aridanesh'
HOUDINI_DESKTOP_DIR                  := '/Users/aridanesh/Desktop'
HOUDINI_TEMP_DIR                     := '/tmp'
SHELL                               := '/bin/bash'
USER                                := 'aridanesh'
Aris-MacBook-Pro:Resources aridanesh$
```



# Common Environment Variables

## ‣ **\$FPS**

- This variable is used by iplay to set the default frames per second. It is automatically set by the OpenGL output driver.

## ‣ **\$HFS**

- The path where Houdini is installed. Houdini reads the configuration information from \$HFS/houdini.

## ‣ **\$HOME**

- Users Home Directory

## ‣ **HOUDINI\_PATH**

- The path of directories where Houdini looks for configuration files.

## ‣ **JOB**

- This variable is often used as a path to the current job. It is not used by any of the default Houdini paths.

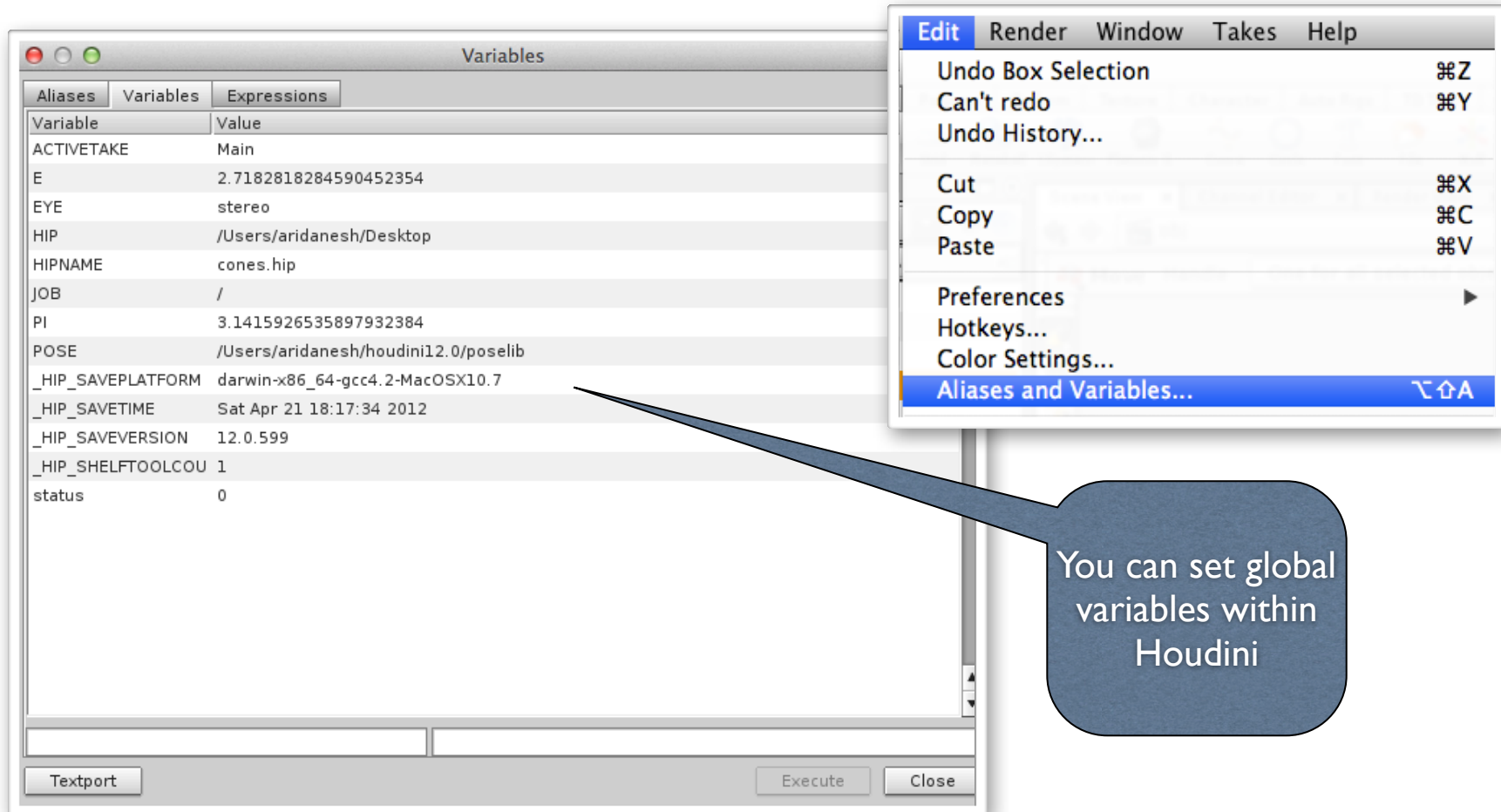


# Global Variables

- `$E` - The mathematical constant  $e$  (2.71828...).
- `$HIP` - Job directory. This defaults to the directory where you started Houdini. You can set it through the Textport.
- `$HIPNAME` - The name of the current .hip file.
- `$PI` - The mathematical constant  $\pi$  (3.1415926...). You can use it to calculate the lengths of arcs.  $2 * \$PI * r$  (radius of circle) equals the circumference. Also,  $\pi = 180^\circ$  expressed in radians.
- `$WEDGE` - Contains the wedge name when using the Wedge ROP.

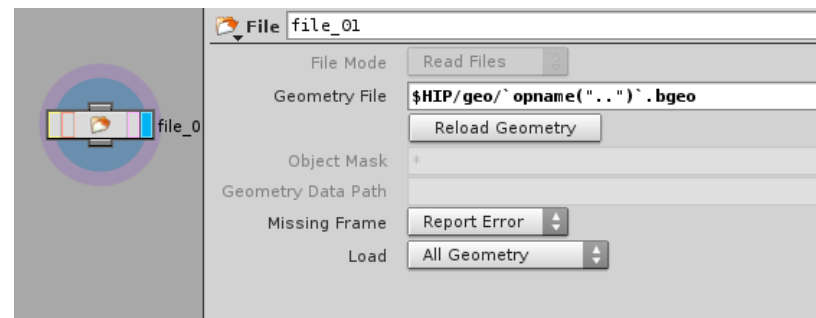
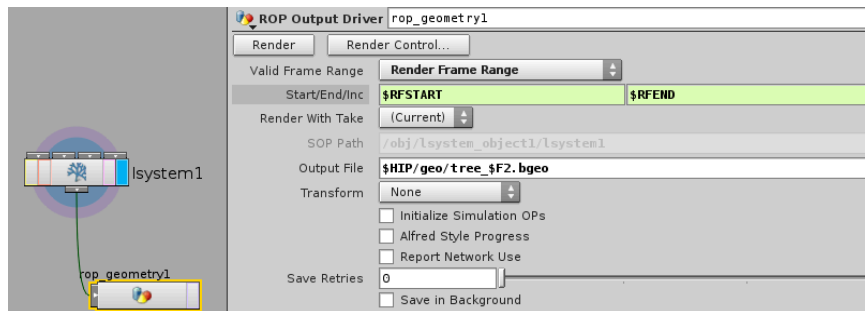


# Global Variables - Settings Inside Houdini



# Path Example - Loading Sequential Files

- Copy and Paste automatically sequential files





# Playbar Variables

- \$FPS - Playback speed in frames per second (as set with the Playbar controls).
- \$FSTART -Frame number of the first frame of animation (as set with the Playbar controls).
- \$NFRAMES (the number of frames in the animation) =  $\$FEND - \$FSTART + 1$ .
- \$FEND -Frame number of the last frame of animation (as set with the Playbar controls). See also \$RFEND below.
- \$F -The current frame, (as set with the Playbar controls). This is a very useful variable, especially for rendered picture filename numbering.
- \$FF - Floating point frame number.
- \$NFRAMES - Number of frames in the animation.  $\$NFRAMES = \$FEND$  (the last frame of animation) -  $\$FSTART$  (the first frame of animation) + 1.



## Playbar Variables (cont.)

- `$RFSTART` - Frame number of the first frame shown in the playbar. The playbar can show a subset of the total number of frames, allowing you to focus on a particular section of a long animation. `$RFSTART` and `$RFEND` control the subset of frames shown in the playbar.
- `$RFEND` - Frame number of the last frame shown in the playbar.
- `$T` - Current time in seconds. Equals  $(\$F-1)/\$FPS$
- `$TLENGTH` - Total length of animation in seconds.
- `$TSTART` - Start time of animation in seconds.
- `$TEND` - End time of animation in seconds.

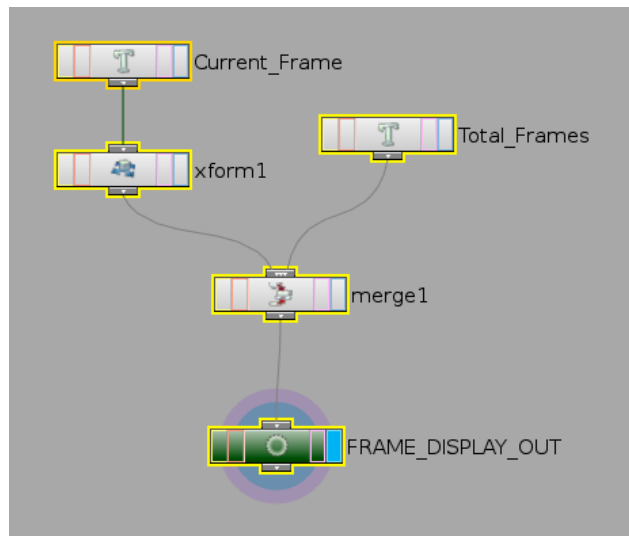
# Playbar Project - Display Timecode

## ▸ Current Frame

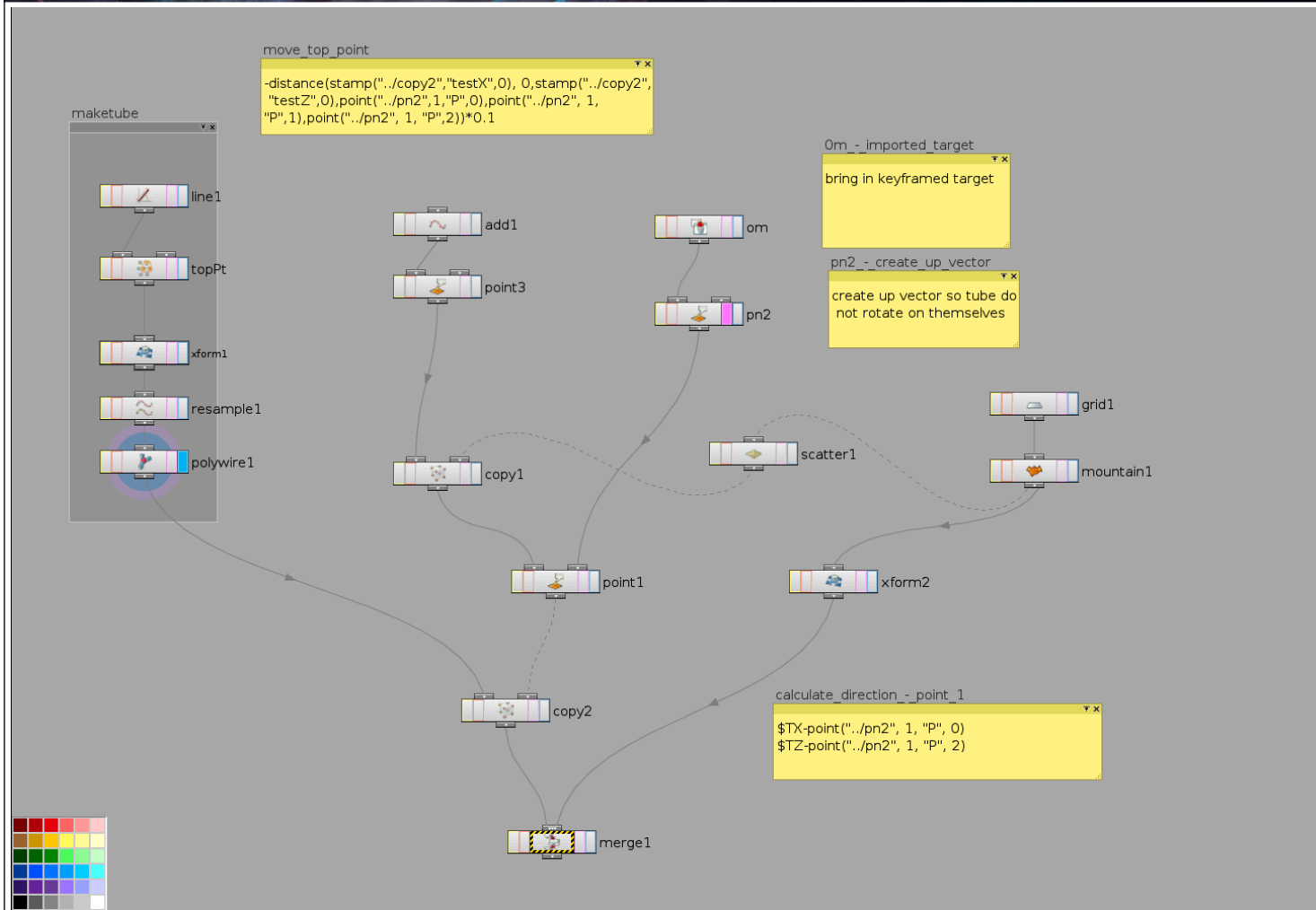
▸ ``padzero(2,(((($F-1)+ch("frameOff"))/$FPS))`:`padzero(2,($F+ch("frameOff"))%$FPS)``

## ▸ Total Frames

▸ `/`padzero(3,($NFRAMES/$FPS))`:`padzero(2,($NFRAMES%$FPS))``



# Project Stretchy Hose





# Key Expressions

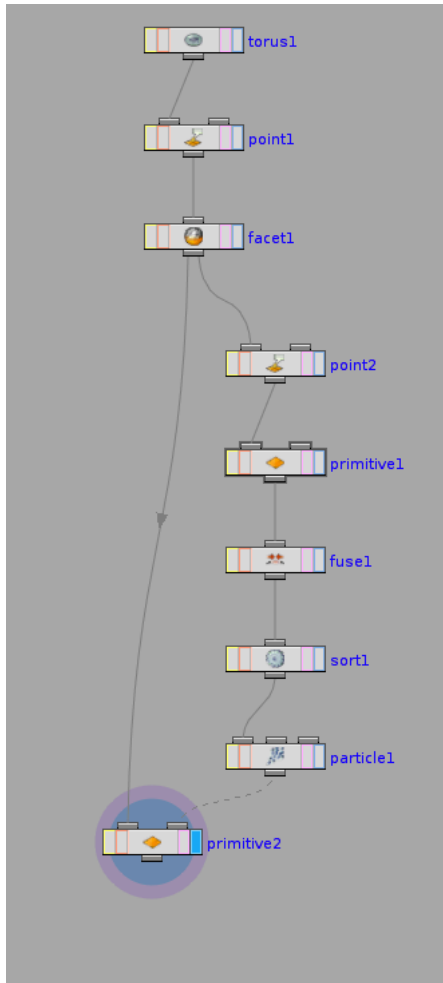
- **Calculate Direction**

- `$TX-point("../pn2", 1, "P", 0)`
- `$TZ-point("../pn2", 1, "P", 2)`

- **Move top point**

- `-distance(stamp("../copy2","testX",0), 0,stamp("../copy2", "testZ",0),point("../pn2",1,"P",0),point("../pn2", 1, "P",1),point("../pn2", 1, "P",2))*0.1`
-

# Project - Unwrap Torus



- Using particles, prims, and expressions to unwrap a torus