Learning Houdini

- When I was learning Houdini, most of the tutorials out there just showed how to do stuff
- I really only 'got' Houdini once I understood how it worked internally
- Learning Houdini is like learning a language with relatively simple, consistent grammar, and a large vocabulary
- Lots of different nodes and you wont understand most of them at first (there are still plenty that I've never used)
- If you know how it all fits together, you can figure out what all the different nodes do, bit by bit
- After this session, you may not be able to **do** much directly, but hopefully it will make sense enough to keep learning

Why use Houdini?

- Different to other apps
- Procedural workflow
- Complexity not something to be scared of, can utilise it easily
- You can do anything, and do it 10 different ways
- Lots of control, everything is accessible

History

- Made by SideFX in Toronto. Private company, around 50 employees
- Branched from Prisms, cousin to Touch Designer
- Used to be only used by smaller teams of specialists doing FX, now most major film productions will have some amount of Houdini usage
- Now making inroads into other disciplines/industries (eg. games/realtime)
- Professional/Indie/Free versions
- Excellent support, nightly builds, public changelogs, access to developers

Workflow Overview

- Don't make a thing, make a system that makes things for you
- Invest a bit more setup time at the start, for huge benefits later on
- Node networks that are actually meant to be used
    - Richer data means simpler data flow
    - Look inside built-in nodes
    - Can follow the chain along and understand other setups (with annotations)
    - Becomes self-documenting
- Consistency and integration
    - Pass data in and out of different networks
    - VEX language used for geometry, shaders, fcurve manipulation, compositing,

User Interface

- Main window divided into tiled panes, resize and subdivide as you like
- Any pane can be any type of editor
- 3D View
    - Hold space to temporarily switch to camera mode
- Network View
    - Shows operators (nodes)
    - Tab to open operator menu. Type to search by name
    - *Display Flags* to control visibility (and other things)
- Parameters
    - Shown for selected node (not necessarily the displayed node)
    - Middle mouse button for multi-resolution 'ladder'

Operators and Networks

- *Operators* (nodes) live within *Networks*
- Networks can be nested
  - Either with subnetworks, to organise
  - Or different networks can be encapsulated inside each other
  - Represented like a unix file system: eg. /obj/my_geometry_object/color1
- Operators have *Parameters* to control how they work
- Operators specific to a certain network type
  - Not like Maya Hyper-etc where everything is mixed together
  - Specific networks for dealing with specific types of data
  - But data can still be passed in and out of different networks

Operator/Network Types

- Several different kinds of operators. A little overwhelming at first, some more obscure than others
    - Objects (OBJs)
    - Surface Operators (SOPs)
    - Render Operators (ROPs)
    - Channel Operators (CHOPs)
    - Shading Operators (SHOPs)
    - VEX Operators (VOPs)
    - Dynamics Operators (DOPs)
    - Compositing Operators (COPs)
    - This sounds crazy! (It's not that bad, you'll be part of the cult soon)

Objects vs Geometry

These are the networks you'll be using most:

- Objects (~ Transform Node)
  - Objects are containers
  - Transform parameters for transforming object as a whole
  - Not so convenient for procedural manipulation en masse
  - Shader assignments, render visibility, etc
- Geometry (~ Shape Node)
  - Surface Operators create, manipulate geometry inside an object
  - The output of a SOP network is the geometry that an object represents
  - Each SOP takes geometry in, does something, sends geometry out all the way down until whatever node is marked as the output
  - Great for handling huge amounts of geometry

Point / Primitive Geometry

- Point: A position in 3D space
  - In Maya, like a vertex (or particle, curve CV, NURBS surface CV, ...)
  - Houdini Point is generalised and consistent
  - No matter what type of geometry you can operate on points in exactly the same way
- Primitive: The 'physical' surface geometry that you can see
  - In Maya, like a polygon face, except...
  - There are **many** types of primitives, not just polygons
  - Polygons, Polylines, Curves, Metaballs, Volumes, and more...
  - Consistency
- You can see what geometry is coming out of any SOP, by middle mouse button on the node
- Any primitive types can live happily side by side (eg. *Merge* SOP)

Vertex Geometry

- Vertex: The link between Points and Primitives
- Primitives are connected to Points via vertices
  - Eg. Primitive A has Vertex B as one of it's corners
  - Vertex B connected to point C, to get its position in space
- Vertices are unique per primitive
- Usually don't have to deal with them that much
- Just remember, Houdini's *Vertices* are different to Maya's *Vertices* - think *Points* instead
- This will become more clear soon

Data Flow Recap

- Each time the scene is *cooked*, it processes the operators from the top, leading to the output
- The parameter values on each operator controls how it is processed during that particular cook
- All relevant geometry data flows down that one wire
- Easy to follow the flow of data
- Some nodes have multiple inputs, to do things with multiple streams of geometry, eg.
    - Merging them together
    - Using one as a reference (copying things from geometry A to geometry B)

Groups

- A way to mask out certain points/primitives/etc
  - To restrict the effects of some SOPs
  - Can be used to isolate and partition bits of geo
- Rough Maya analogy: Selection Sets?
- Groups are binary on/off, a piece of geometry either belongs to a particular rgroup or it doesn't
- Geometry can belong to more than one group at a time
- Groups can be definited by point/primitive numbers, but better to be procedural!

Attributes

- By default, Points represent just positions, Vertices/Primitives just connectivity, but...
- Points/Primitives/Vertices can have **any** other type of information associated with them, flowing in the same stream of geometry
- These are called *Attributes*
- Normals, UVs, point colours, whatever: they're all attributes.
- Attribute types:
    - Float/Vector/Integer/Matrix, etc etc
- Everything you need to construct the scene is accessible in the spreadsheet

Modifying Attributes

- That's what SOPs do!
- Some do it very explicitly, like *Attribute Create*
- But in general, each SOP just reads in geometry and attributes, modifies them, and spits out a new stream
- Once you get into that mindset, you can even think of Houdini as a super fast glorified spreadsheet editor, with a 3d view
- And that if you have a generalised way of manipulating attributes, you can do whatever you want

Using VEX

Oh no, coding! Dammit, I knew Houdini was for nerds!

- No, it's actually quite simple, accessible, and consistent
- It's what really opens up Houdini and makes almost anything possible
- Very useful for short little snippets, gluing bits together, not always big monolithic tools (i.e. Maya plugins)
- From Renderman Shading Language era heritage
    - SIMD: Single Instruction Multiple Data
    - Very much like OpenGL Shading Language
- Operates on a single Point/Primitive/Shading Sample/Pixel/etc, but on all at once
- Multithreaded by default, uses all available CPUs very well

VEX Wrangle SOP

Wrangles are an easy way to use VEX to modify geometry and attributes

@ (at) syntax: Modifying an **At**tribute

- Set P attribute to the coordinates: x:0, y:1, z:2

  ```
  @P = {0, 1, 2};
  ```

- Set the colour (Cd) attribute to be equal to the normal (N):

  ```
  @Cd = @N;
  ```

- Raise all the points up in Y by one unit

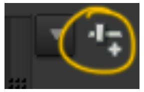
  ```
  @P.y = @P.y + 1;
  ```

Vex Wrangle SOP

Going further…

- Inflate the points by pushing their positions (P) along the normal (N) (Vector addition, remember high school maths!)

```
@P += @N;
```

- How do we make this more tweakable? Use chf() to read in the value of a float parameter. The button next to the code editor conveniently creates any referenced parameters on the wrangle SOP: 

```
@P += @N * chf("push");
```

- Hundreds of VEX functions, lots more that you can do

VEX Operators (VOPs)

- A node interface to work with VEX
- Generates VEX code on the fly
- A bit easier to remember what nodes are available (and their inputs)
- Can get complicated with lots of nodes and complex logic - sometimes wrangles are quicker and simpler
- Same concept as wrangles, working on a single component, but in parallel
- Data flow left to right
- To read and write attributes:
    - *Bind* VOP
    - *Bind Export* VOP
- To add a tweakable parameter:
    - *Parameter* VOP
    - Middle mouse on input - Promote Parameter

Example: Engineer setup

- Using built in mocap biped
- Scatter points on geometry
- Distort those points using noise

Simulation

- Recap: Houdini cooks top to bottom, from scratch, every frame
  - Well, only if it actually needs to
- Time dependencies
  - If geo is unchanged it won't cook it each frame
  - Only will re-cook nodes downstream from the point where they start changing over time

Dynamic simulation

- Usually requires the previous step's data, to work progressively
- How do we do this if it's being re-created from scratch each frame?

Particles (DOPs)

- DOPs works iteratively, based on previous state
- A little different to SOPs, not purely geometry data flow - you also defines relationships between simulation objects
- POP Object
    - Simulation object (a container for DOPs relationships etc)
- POP Source
    - Add particles to the POP Object, from SOP geometry
- POP Solver
    - Controls the updating of positions and velocities of the POP object data
- Merge
    - Define collision (or other) relationships

Many other different types of DOP solvers. Can make your own, too!

Rigid body dynamics

- Bullet library
- Uses *Packed Primitives*
- Packed primitives let you work with single point representations.
  Consistent with POP data, can use similar POP forces
- RBD Packed Object
- Rigid Body Solver

Audio reactive animation

- Read an audio file in to CHOPs
- Filter the channels as we need
- Export that back out as attribute values on geometry
- Then we can drive whatever we want with those attributes

CHOPs

- CHOPs: Channel Operators
- A bit weird and obscure, due for a refresh in future versions of Houdini
- *Channel*: 1D data, animation of a single value over time
- Rather than evaluating at a single point in time, CHOPs deal with the timeline as a whole
    - To do this, it must evaluate the entire time range first so it's all available in memory
- CHOP network can be a subnetwork, or just in the default /ch network.
- It's possible to manipulate the animation curves of positions of thousands of individual points directly in CHOPs
    - But that gets really complicated so we won't do it that way

Getting data out of CHOPs

- *File* CHOP reads in Audio
- Timeline Audio button lets you choose a CHOP for playback
- Use *Null* CHOPs to make convenient markers to import/export
- If stereo, can delete one channel with *Delete* CHOP

In SOPs:

- *Channel* SOP to read in the values of a channel as point attributes
- Static mode maps samples to point numbers
  - Attribute values static over time
- Animated mode animates attribute values on individual points
  - Must have the same number of points as channels in the CHOP

Getting *useful* data out

Very noisy and jittery, and just a single value

In CHOPs:

- *Resample* CHOP to convert to 24 Hz
- Or *Resample* to medium rate, then *Filter* CHOP to smooth it out

But that's just a single channel...

- *Pitch* CHOP to split into channel per frequency band
- *Rename* CHOP to rename the channels into a convention that can be imported easily
- Must have a point per channel

Procedural animation

- Varying degrees of procedural-ness
- Animation with keyframes
  - Manually setting keyframes on existing nodes
  - Can use keyframes on spare parameters to do custom things
  - Blending between states
- Animation with (simple) math
  - Lots of functionality in VEX/VOPs
  - noise!

Animation along curves

- primuv() VEX function, *Primitive Attribute* VOP
- Looks up the value of any attribute (eg. P) given a uv value
- Curves have implicit U coordinate running along their lengths
- Great for visualising and manipulating the animation paths in 3D

Matt Estela's Pages

- Matt is an FX lead at Animal Logic
- Recent convert to Houdini, and has documented his journey very well
- Lots of great little example setups (with GIFs!) to download and play with
- **Highly** recommended

Maya to Houdini guide:

- http://www.tokeru.com/cgwiki/index.php?title=MayaToHoudini

Read **all** of these:

- http://www.tokeru.com/cgwiki/index.php?title=HoudiniGettingStarted

Me

- http://mattebb.com
- matt@mattebb.com