

I Modelling OPerations

I INTRODUCTION TO OPERATIONS

VIEWPORTS

Viewports can be used both to view and to construct geometry. As such, you can invoke Operations within a Viewport to perform a number of related viewing or modelling commands.

OPERATIONS

The icons along the top and left of the Viewport represent the different Operations available. To invoke an Operation, type `(Tab)` and select from the pop-up menu of Operations, or click its icon.

PARAMETERS

The most common parameters for an Operation are displayed along the top of the Viewport. Call up all the Parameters for a Operation by clicking the `+` button.

OPERATION PARAMETERS

In addition to the Parameters available in the Viewport, an OP node often has many more parameters as well – these are displayed when you view the individual OP node parameters (using a *Parameter* editor – as set in the Pane menu), which are displayed when viewing the individual OP nodes in a Network Editor.

OPERATION MENU (`(Ctrl) P`)

Using `(Ctrl) P` within the Viewport displays an Operation menu. This menu is context-sensitive, the available options will change depending on whether you are in Model Mode (i.e. viewing Operations), or simply transforming objects (i.e. viewing Objects).

SUB-ICONS

When you enter a Operation, you will see a set of corresponding Sub-icons and frequently used Parameters along the top of the Viewport. These allow you to quickly set commonly used parameters, for example – selecting the primitive type for a Curve Operation.

I.1 VIEW ONE / ALL OBJECTS

You can work in context of the entire scene, or view only the geometry of the current Operation. Do this by clicking the *View One/All Objects* button at the bottom of the Viewport in the Viewport controls, or typing: `(Space) [E]`.



View One/All Objects

You can pick another object to edit by using the Select Object Operation. While in the Select Object Operation, the *View One/All Objects* button will be disabled (implicitly on).

I.2 SNAPPING

Snapping allows you to build geometry more accurately by making your cursor lock onto elements as you move it. See *Snap & Snap Options...* p. 119 of the *Interface* section for more information.

I.3 LOCKING AN OP

Houdini allows you to lock a OP in two ways: Hardlock and Softlock. A Hardlock prevents a OP from cooking so that manual modelling changes are retained instead of updating parametrically. A Softlock is a special case of unlocking that allows a subset of manual modelling changes to be made while maintaining the geometry's parametric data flow within the OP network. A Softlock can only be maintained so long as the geometry's topology doesn't change (i.e. the number of points).

To lock a OP, click on the Lock icon on the OP's tile (you must use a Network editor pane to view ops first). This Hardlocks the geometry within the OP and prevents the OP from cooking or changing.

A Hardlock is displayed as a yellow lock; while a Softlock is displayed in light blue.

UNLOCKING AN OP – UNLOCK VS SOFTLOCK

You unlock a OP by clicking the *Lock* icon on the OP's tile. Doing so displays a dialog which asks you if you would like to *Unlock*, or *Softlock* the geometry.

Warning! If you select *Unlock*, you will lose the work you did in the Model Editor, and the changes you made manually are over-ridden with new geometry as defined by the parameters for that OP. For this reason, changing the parameters of a Locked OP has no effect; and unlocking a OP gives a warning message.

If you select *Softlock*, then assuming you do not add any points or primitives, the soft-locking will maintain your modelling changes and still allow the OP to be recooked. Sound like magic? It is.

LOCK SAVE AND READ IT

Once you have geometry contained in an OP to your satisfaction, and because the potential exists to lose your model data when unlocking an OP, you are advised to save the geometry to disk (use the *Save Geometry...* option from the OP's pop-up menu) as a *.geo* or *.bgeo* file. Then, follow the locked OP with a File OP to read the geometry back in.

This not only prevents any possible accidental loss of work from unlocking the OP; but is only useful in avoiding waiting for the OP to cook for static, non-deforming geometry.

2 SELECTING

This section assumes that you already know the differences between geometry types. For information on geometry types, see the 13-Geometry Types section.

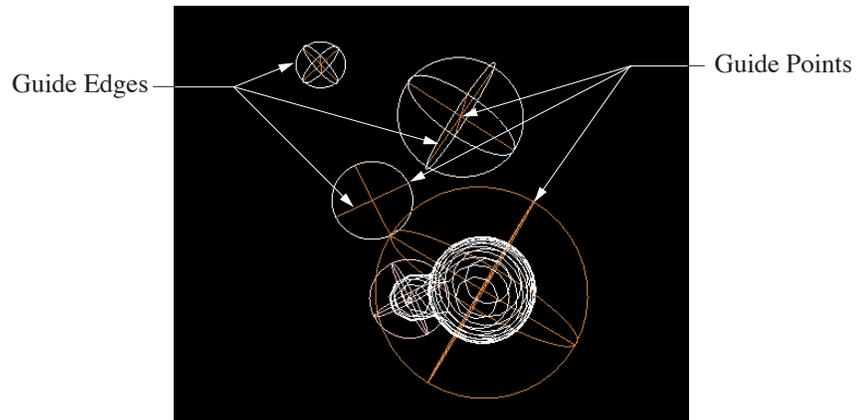
2.1 SELECTING

Before performing any Operation, you need to tell Houdini what to Operate *on*. You do this using the *Select* Operation. You should specify the type of geometry you wish to select before going at the geometry through the icons located along the left-edge of the Viewport. They allow you to specify: points, edges (any segment between two points, including a spline hull), vertices, primitives, point groups, and primitive groups. When you make the selection, it turns yellow.

Note: If you overflow the geometry selection buffer by selecting thousands of points or primitives that you want in one step, then only a subset may actually become selected. The solution is to select the points/primitives in several steps using the **Shift** key to add to the selection.

2.2 GUIDE EDGES AND POINTS

Before discussing selection, it is important to know the difference between guide edges and guide points. A guide edge is a line that doesn't connect two geo points. For example, the radius of a circle. The guide point is the end of a guide edge.

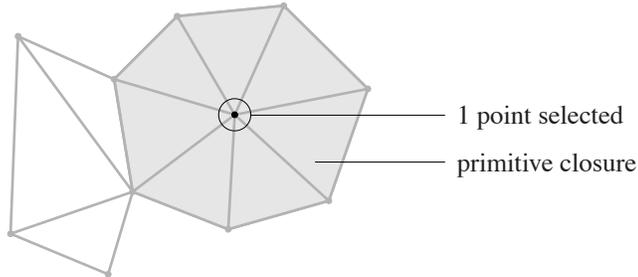


You will see these guide edges whenever you encounter a primitive shape (circle, sphere, tube, and metaball).

2.3 SOME TYPES OF SELECTIONS

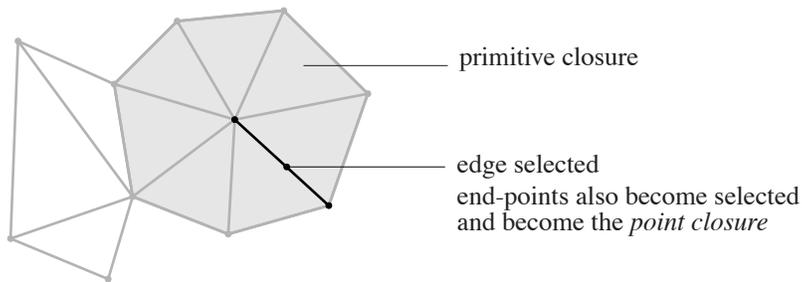
POINTS

If a point is selected, then all primitives that contain that point are displayed in green. This green area is referred to as a primitive closure. The primitive closure highlights the area that is affected by modifying the selection.



EDGES

If you select an edge, that edge becomes selected, as well as the two points which delimit its ends. The primitive closure includes all primitives that share the end points of the selected edge. Edge selection and deletion differs somewhat depending



on the geometry type you are working with. These behaviours are outlined below.

polygon edges

If the curve you are selecting an edge from is closed, and you select and delete some edges the curve will be opened after the deletion.

If the curve is open, and you delete from either end of it, then those edges will be removed (unused points will be eliminated as well). If the curve is open, and you delete from the middle of the curve, then the curve will be split into multiple faces (primitive, point and vertex attributes will be preserved).

bezier edges

Selection and deletion of Bézier edges is similar to Polygons but there is a restriction as to which edges you can delete—you can only delete entire spans from the curve. If one edge in a span is not selected for deletion then none of the span can be deleted – the un-deletable edges will remain selected after the deletion.

nurbs edges

Again, this geometry type is similar to Polygons, but there is a restriction as to which edges you can delete. Regardless of the position of the edges being deleted, there must be at least ‘order’ edges left between any two sets of deleted edges. Thus, if you have a curve of order 4 with 12 edges and you choose to delete the middle 6 then one edge from either end of the selection will not be deleted so that the remaining two curves can have 4 edges each (assuming the curve was originally open).

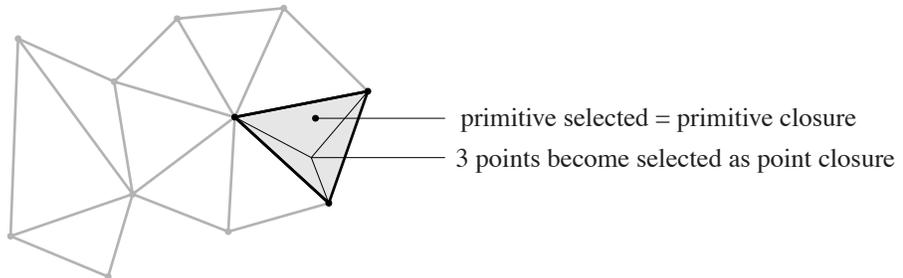
mesh edges

Selecting and deleting a “ladder” (the pieces of the rows of the mesh bounded by a column) splits the primitive into two primitives.

Note: Even though you can open/split spline types with edge deletion it is not terribly useful if only because of the restrictions. It is recommended that you split/open a spline curve using the Carve Operation instead – you’ll have much better control over the results.

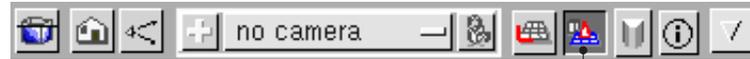
PRIMITIVES

If you select a primitive, the points contained by the primitive will form the point closure. The points used by the primitive closure also become highlighted.



2.4 VIEW SCENE / OBJECT

Enabling the *See One/All Objects* button at the bottom of the Viewport allows you to quickly view and select another object while you are modelling. Use the volatile key (*Space* *E*) to quickly view all objects and select another object for editing.

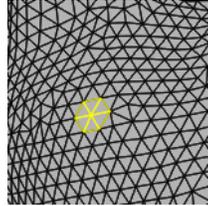


View One/All Objects

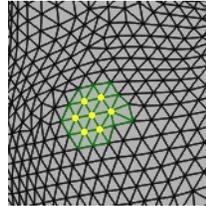
Simply click on the object you want to edit, and it becomes the current object. You will see the path (e.g. */obj/geo3*) update in any Network Editor or Parameters panes.

2.5 GROWING THE SELECTION

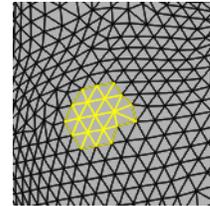
Once you have made a selection of points, you may want to ‘grow’ the patch of polygons you’ve selected along its boundary. You can implicitly do this by alternately typing **P** (select Points) and **F** (select Primitives), like so:



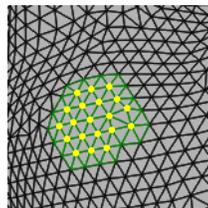
Selection



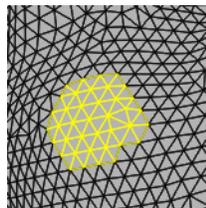
Type **P**



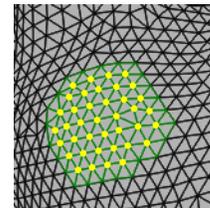
Type **F**



Type **P**



Type **F**

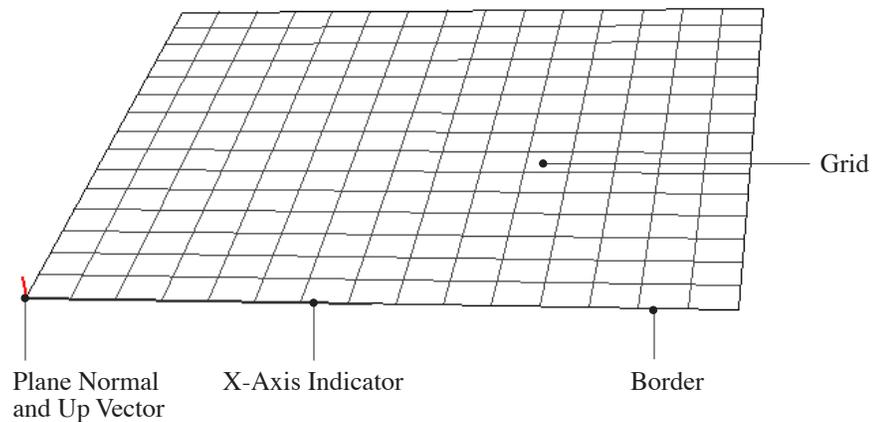


Type **P**

3 CONSTRUCTION PLANE

To construct geometry in Model mode, you need some point of reference against which to draw. In the Model Editor, this is the *Construction Plane*.

The Construction Plane is your primary point of reference for constructing geometry in the modeller. It initially appears as an XY coordinate grid whose size, origin, and orientation can be changed to align with existing geometry.



The components that make up the Construction Plane are described in detail below.

THE GRID

The grid provides you with a sense of scale as you create your geometry.

The grid of the Construction Plane is initially made of twenty × twenty squares of 0.1 unit size each. Change its size with the Construction Plane pop-up menu () > *Increase Grid Count* and *Decrease Grid Count*. The keyboard short-cuts for these two operations are: and . Also see the *Construction Plane Operation* - p. 507.

An active grid (sea blue) indicates that something is being constructed relative to Construction Plane. When the grid is inactive (for example, when the View Operation is active) it becomes grey.

THE BORDER

The plane's border is grey, and acts as a rubber-band to indicate the extent of the Construction Plane. It stretches to indicate when the geometry you are dragging is moving outside the current bounds of the grid. This feature gives a good indication of relative distance in 3D perspective.

PLANE NORMAL

The plane normal is red, and is located at the origin of the plane, which is at the lower-left corner. It is always perpendicular to the Construction Plane (and indicates the "direction" it faces).

UP VECTOR

The Up Vector is pink, and initially coincides with the Plane Normal. Therefore, initially, it is not visible. The Up Vector indicates the direction of translation when you **Alt** - Drag geometry. It normally points in the same direction as the plane normal, but can be changed (see *Up Vector Operations* - p. 509).

X-AXIS INDICATOR

The X-Axis indicator is red, and extends from the plane origin to half the length of the border along the local X-Axis of the Construction Plane. It provides you with an indication of the Construction Plane primary axis.

3.1 SHOWING / HIDING THE CONSTRUCTION PLANE



click here to Show/Hide Construction Plane

3.2 MOVEMENT ON THE PLANE

Often, geometry must be moved relative to the Construction Plane. To select and move geometry, click and drag it along the plane. Constrain cursor movement to 45° increments by holding down the **Ctrl** key. This constraint is indicated by a red-dotted line on the Construction Plane, and is relative to the point when **Ctrl** was depressed.

MOVING UP

To move things in the direction of the Up-Vector, hold down the **Alt** key and drag. The object will move up (or down) in the direction indicated by the pink Up Vector. This displacement begins directly above the cursor's *current* XY location, and not necessarily from where you first started dragging the object.

If the geometry being moved is not on the Construction Plane, it is tracked by a small grey cross-hair. The arm of the cross-hair indicates the X-Axis of the Construction Plane. The size of the cross-hair is one grid unit.

There is also a small Projection Vector which drops perpendicularly from the object being dragged, down to the cross-hair on the Construction Plane. A portion of this Projection Vector is red, and the rest of it is grey. The red portion shows the delta (change) that you've made in the object's elevation, while the grey portion shows the initial elevation of the object relative to the Construction Plane.

CHANGING THE UP-VECTOR



To change the Up-Vector, you need to enter the Construction Plane Operation via the icon in the Tool-Bar. To define an up vector, enter three points with the middle mouse

() to define a flat surface. The up vector is then changed to be perpendicular to the imaginary surface created by the three points.

There are many ways to define the up vector. See *Up Vector Operations* - p. 509.

3.3 GEOMETRY AND THE CONSTRUCTION PLANE

Most of the modeling Operations presented in the following chapter require the presence of a Construction Plane. For example, circles, NURBS curves and metaballs are built on it (or at a fixed elevation relative to it); primitives, edges and points are translated on the plane. Although it is usually described as the foundation of geometry creation and manipulation, there are cases that over-ride the Construction Plane's priority: for example, snapping may cause a circle to be built on a spline curve or on a surface if *Snap to primitives* is turned on. Similarly, the direction of the up-vector may be completely unrelated to the orientation of the plane, in which case if the  key is held down when building the circle, the circle's X radius will be aligned to the up-vector.

3.4 CONSTRUCTION PLANE OPERATION

Please see *Construction Plane Operation* - p. 507.

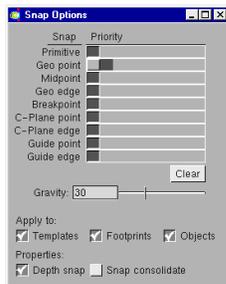
4 SNAPPING TRICKS

4.1 INTRODUCTION

The following are examples of how to use some of the modeller's snap tools.

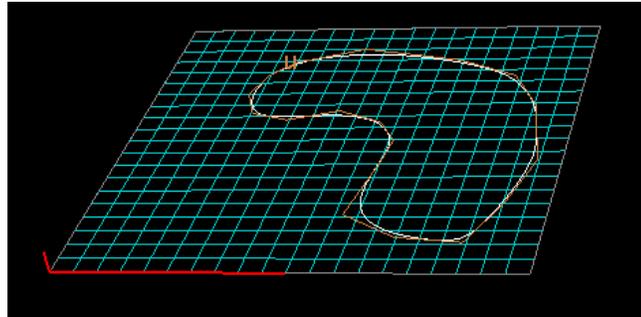
4.2 DEPTH SNAP

Display the Snap Options by selecting *Snap Options...* from the Snap-to pop-up menu at the top-left of the Viewport while editing *Geometry*.

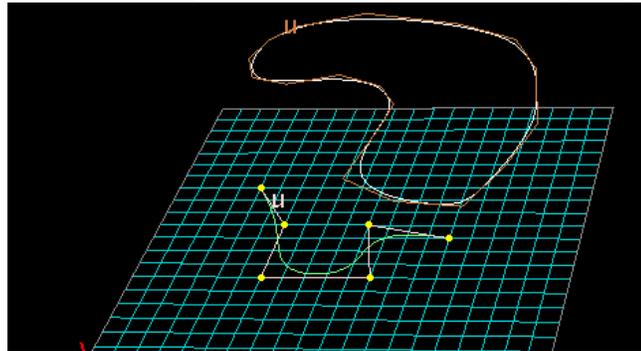


The *Depth Snap* option is handy when you want to copy the outline of primitives that are located in different planes. It will find the location of the X, Y and Z of the snap point. This is the normal setting when the snap option is first initialized, and is what you would normally expect when you want to snap. However, when this option is turned off, you are drawing on the Construction Plane and snapping to a point that is projected onto the plane along the Construction Plane's normal.

For example, suppose you started with an outline shape and then moved the Construction Plane down in the Z direction as illustrated below:

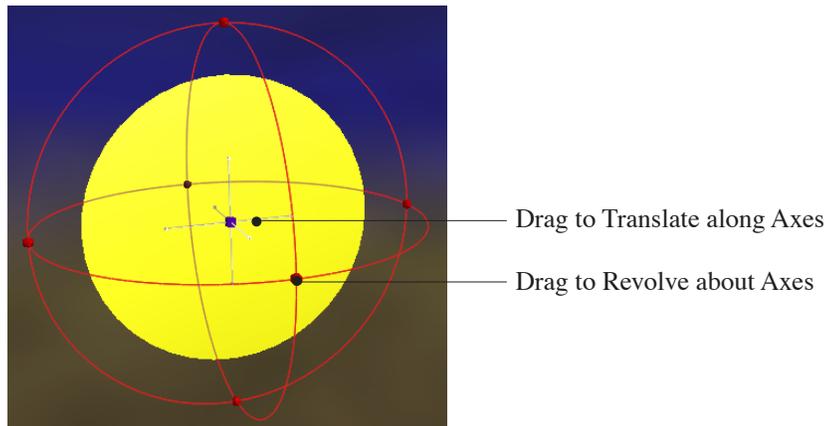


For your second shape, you want to follow a few points from the original shape, a few of the hull edges, and a few of the Construction plane points:



When you start to pick a point to draw, the snap option overrides your point click if you are within the Gravity radius (measured in pixels) and you will see that, even if you click on a point which is on the original shape, you will still be drawing on the Construction Plane.

5 HANDLES



There are handles associated with most Viewport Operations – these are specific to the Operation (i.e. Select, View, Group etc.).

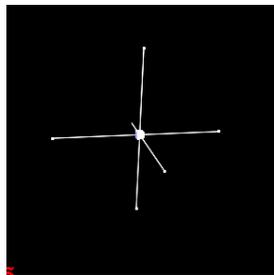
TRANSLATION / ROTATION / SCALE

The *Handle* is a device that allows you to translate, scale, and rotate the selected elements by manipulating the small nodes along it.

You can decouple the handle from the geometry by typing or selecting *Toggle Handle-Geometry Detachment* from the pop-up menu). This is very good for initially placing the handle for a proportional scale, or if you want to snap to some geometry with some reference point. Then couple the Handle back together with the geometry to manipulate it in reference to that geometry.

TYPES OF HANDLES

The three basic types of Handles used in Houdini:



Translation Handle



Rotation Handle



Scale Handle

MAKING HANDLES PERSISTENT

Clicking with the on a Handle yields a pop-up menu from which you can select an option called *Persistent*. This makes the handle stay there, even while you go on to do other things. You can view and manage a list of persistent handles by selecting from a Pane's *Pane* menu > *Handle List*.

5.1 HANDLE MENU ()

TRANSFORM HANDLE PARAMETERS

translate handle ()

For moving things around.

rotate handle ()

For rotating the selection.

scale handle ()

For changing the size of that which is selected.

move pivot to world origin

Moves the pivot about which transformations occur to the world origin.

move pivot to cplane origin

Moves the pivot about which transformations occur to the origin of the Construction Plane.

project pivot onto cplane

Projects the pivot onto the Construction Plane.

DISPLAY HANDLE

When enabled, it displays a handle (one of: Translation / Rotation / Scale) with which you can manipulate the selected geometry. You can set which type of handle is displayed using the options (  ), above.

Note: See *Handles* p. 408 for a description of the available manipulator Handles.

PERSISTENT

Makes the handle persistent, so you can go on to tweak other parts of the geometry with additional handles. Persistent handles will show up in an *Pane > Handles List* pane.

HANDLE PARAMETERS

Displays a dialog with parameters applicable to the current handle.

OP PARAMETERS...

Displays a dialog with parameters to allow numeric control of the handle. See *Parameters – Transformations* Page p. 810, below.

ANIMATABLE PARAMETERS...

From the list of parameters (see above), this will display a list of those parameters to which you can add a channel and animate. Click on the small 'scope' icon to the left in order to add a channel in which you can specify keyframes.

STEP TO PREV / NEXT KEY

When a handle has been animated, these two commands allow you to step between an animated handles' keyframes.

SET KEYFRAME

The control and positioning of a handle can be animated over time using keyframes. After you have set your handle in position for a particular frame of your animation, you will need to 'keyframe' it with this command.

REMOVE KEYFRAME

In case you no longer want a keyframe which defines a handle to be at a particular setting at a particular frame.

SCOPE APPEND CHANNELS

Allows you to view the Graph of the handle's keyframed channel.

Unlike the command below, this will add the graph of the particular channel into the mix of channels already being displayed in the Channel Editor.

SCOPE CHANNELS

Allows you to view the Graph of the handle's keyframed channel in the Channel Editor. Previously displayed channels will be replaced with a display of the current channels only.

DELETE CHANNELS

Removing a keyframe deletes a single defining point in a channel. This command gets rid of the channel which holds variations over time altogether.

LOCK / UNLOCK PARAMETERS

You may wish to lock a parameter so you don't accidentally change that perfectly tuned motion you just worked out.

REVERT TO DEFAULTS

Back to the default settings – you can redefine your own defaults.

REVERT TO AND RESTORE FACTORY DEFAULTS

If you have made default settings, but you want to go back to a total reset, this is it.

MAKE CURRENT VALUES DEFAULT

Allows you to set your own default values.

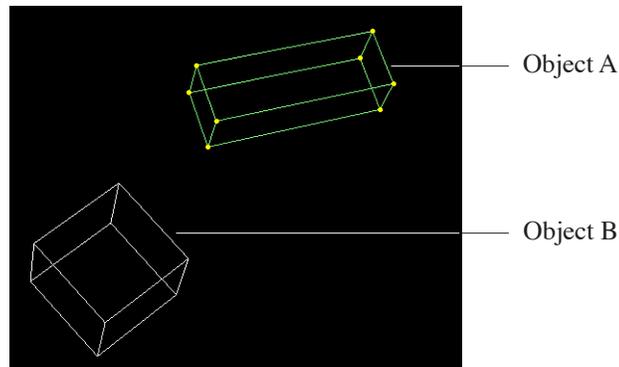
5.2 SOME HANDLE TECHNIQUES

We look at several techniques in which we can use the Transform Jack to align object faces, move objects along another object's face, and align without translating.

ALIGNING OBJECT FACES

step 1 – select and lock

To align an object's face (Object A) with another object's face (Object B), first you must select Object A with a Select Operation (type $\overline{\text{Tab}}$) and secure that selection by clicking the *Secure Selection* button – this ensures that further clicking in the scene doesn't change the selection.



step 2 – setup to transform

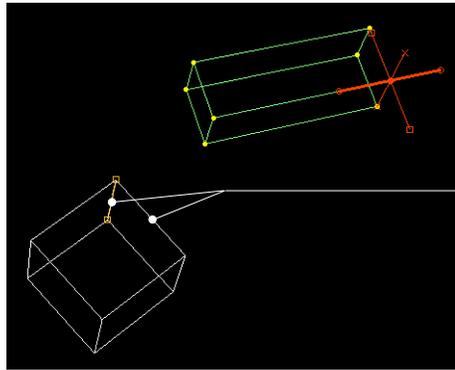
Next, use a Transform Operation (type $\overline{\text{Tab}}$), and then select *Transform*), select the object and F9 to confirm – then select a Transform Handle by typing the $\overline{\text{T}}$ key.

step 3 – orient handle

You temporarily want to decouple the geometry from the handle, so type ' (or select *Toggle handle-geometry detachment* from the F9 menu), and then select *Start Orientation Picking* by selecting that option from the F9 handle menu – the two edges that you click will orient the handle to that face.

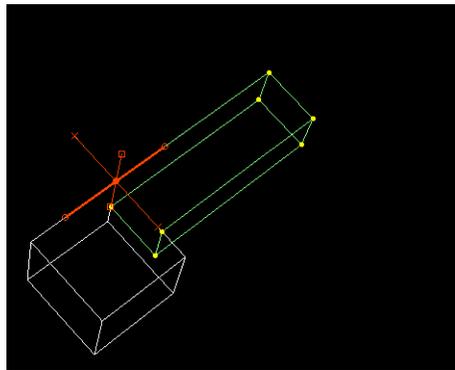
step 4 – align faces

Reenable the handle-geometry detachment by typing ‘ (or selecting from the menu). Since the geometry is now coupled with the geometry, orienting the handle again will move the geometry along with it. So, select *Start Orientation Picking* again from the  menu, and click on two adjacent edges of Object B’s destination face.



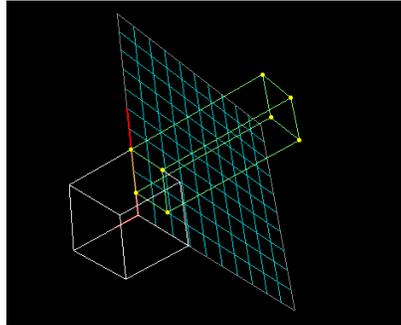
Click here with , and Object A aligns with this face of Object B

This moves Object A and its Handle to Object B’s face:

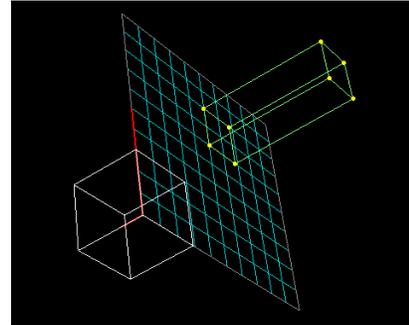


Tip: Instead of clicking on adjacent edges, you can also click on one edge and on the non-shared end-point of the other edge, or on the three edge endpoints.

5.3 OTHER TECHNIQUES



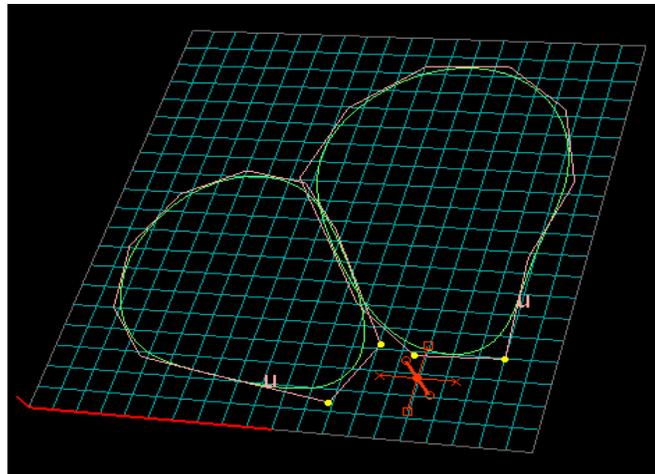
Object A aligned with Object B's face



Construction Plane aligned with Object B; after dragging

LOCALISED TRANSFORMATION OF SELECTED POINTS

To transform only the points in a selection (even of different geometric entities), box select them in the Select Operation with *Point* type selection; then use a Transform Operation to get the Handle; and then any manipulations of the Handle provide a localized transformation of a subset of all points included in the selection.



For example, in the above illustration, several points in the hulls of two NURBS circles were selected, and simultaneously deformed by manipulating them with a Transform Handle.

6 PASTING

6.1 INTRODUCTION

Pasting is a revolutionary surface refinement tool that adds NURBS and increasing the complexity of the base surface.

BENEFITS OF PASTING

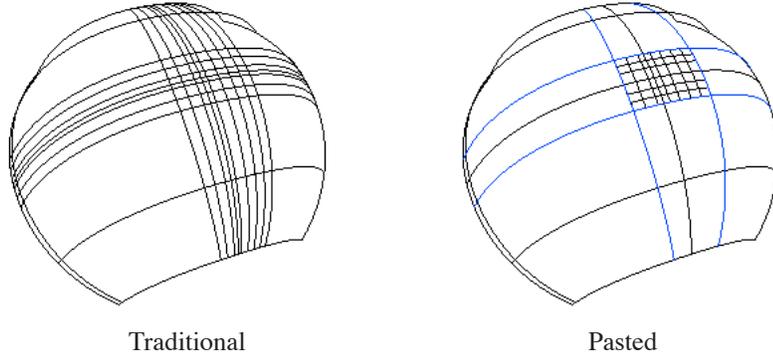
Surface pastings benefits include:

- Low cost, space-efficient, refinement mechanism with local detail and deformation control.
- Interaction with the surfaces in real-time as a whole, or with each surface individually both before and after the surface is pasted. This flexibility in editing geometry allows for rapid prototyping and experimentation at any stage of production.
- Records the surface composition as a dynamic hierarchy – just animate the base and the features will follow.
- Conversion of an entire paste hierarchy into a single surface: polygonal or spline-based. Treating a paste hierarchy as a single surface allows you to simplify and easily integrate the new geometry into your project.
- Ability to repeat the paste procedure on already pasted geometry, and have Houdini treat the resulting geometry as if it was a single surface.
- Works transparently with NURBS and B-splines.
- Encourages reuse of parts on other projects, allowing you to construct a library of already-modeled geometry.
- Pasted detail can be added either by dropping an existing feature onto the base or by extracting a sub-surface from the base. A surface derived from the base can be then set aside for future use.
- Pasting is not a restrictive procedure – paste the feature onto the base either upwards (as a bump), or downwards (denting the surface).
- System-level integration with other surface properties, including texture, color, and transparency.

DIFFERENCE BETWEEN PASTING AND TRADITIONAL REFINEMENT

The main drawback of traditional refinement methods is the often unnecessary addition of control vertices, leading to heavy geometry. Unlike polygonal meshes, which can be refined by subdividing only the areas where more detail is needed, spline surfaces require the insertion of entire rows or columns. Some of the resulting vertices are then displaced to model the intended detail, but the others are simply excess baggage.

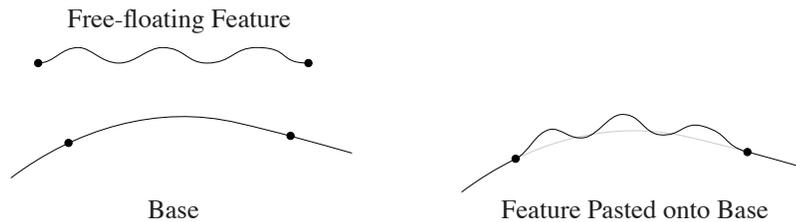
Traditionally, the more detail required, the denser the spline surface will become. If an area already refined needs further detailing, the refinement process will repeat and, soon, the surface will become an unwieldy mesh of points.



Ideally, one would want to add the detail as a seamless displacement from the base surface, leaving the base untouched. If more detail is needed, it should be added to the composition as if the composition were a single surface. These are the primary goals that Houdini's latest modeling tool, *Pasting*, has been designed to meet.

6.2 DISCUSSION

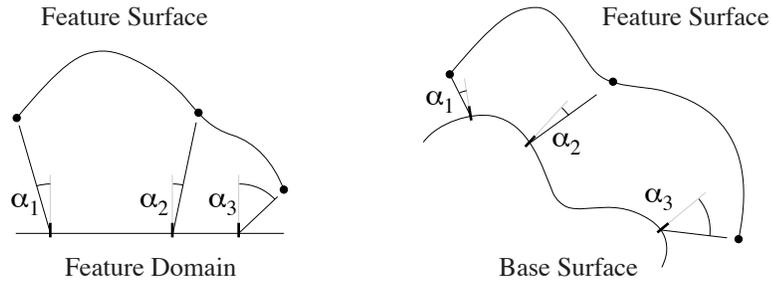
Pasting uses an intelligent displacement technique that embeds the underlying domain of the detail surface into the domain of the base surface. It moulds the vertices of the detail onto those of the base in a seamless fashion. The resulting structure is a paste hierarchy, also known as a multi-resolution surface, consisting of multiple NURBS and/or B-splines.



The feature can be a spline surface or an entire paste hierarchy. If it is a hierarchy, the molding of its root surface onto the base composition will trigger a recursive reshaping of all its pasted elements until the whole feature hierarchy has been mapped onto the base.

The shape of each feature is highly dependent on the shape of the underlying surfaces. Because the feature vertices are computed as offsets from the base surface along its normals, the higher the curvature of the base, the greater the distortion of the feature. Therefore, the angle between normals plays an important role in the amount of feature deformation.

Whether applied to a flat, narrow area or to a large, wavy portion of the base, the feature moulds itself onto the base in the most natural way.

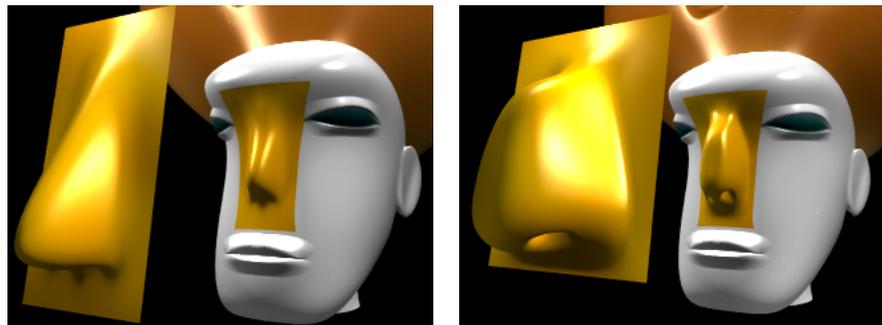


Houdini's pasting operator (Paste SOP) offers excellent control over the shape of the pasted feature. Furthermore, Houdini's architecture allows modelers to alter the look of the detail even after it has been pasted, without loss of continuity.

Because the paste hierarchy is both an organic structure as well as a cluster of regular spline surfaces it offers both flexibility and power to our approach. Thus, if one or more base surfaces deform during modeling and animation, all of the features pasted entirely or partially on them will react to the change accordingly. Almost any modeling operation in Houdini can be applied to the pasted surfaces as if those surfaces were isolated, without the risk of damaging the hierarchy. For example, one can refine, reparameterize, subdivide, raise the degree, fillet or trim a feature while pasted. Of course, Houdini's procedural paradigm allows you to perform these operations before pasting as well.

The pasted features can overlap and slide under each other freely. They do not need to be aligned parametrically to the domain of the root surface, but must be fully contained by it. Furthermore, as features slide, become unpasted, or deleted, their children automatically repaste onto the remaining composition in an inherently organic fashion.

Unpasted features, such as noses, limbs, muscles, or door handles, can be repasted onto other hierarchies later. The reuse of parts becomes a natural modus operandi that cuts down the modeling time while encouraging rapid prototyping. Adding personality to a face, for example, simply becomes a matter of choosing the most suitable features from a library of already-made body parts.



Feature and base surface experimentation with reuse of parts

Another advantage of pasting is that features can be pasted both positively, to create bumps, or negatively, to create dents, simply by flipping the orientation of the vertex displacements. Houdini provides an easy mechanism to remove that part of the base surface where the overlay has been applied.

Since the pasted surfaces are placed in a hierarchy, many graph operations apply. For example, it is possible to replace one or more pasted surfaces with another hierarchy, to change the parents of a pasted surface, or to insert a new hierarchy between already pasted surfaces.

If a seamless connection between feature and base is not required, the pasted detail can be lifted off the base along its normals and allowed to levitate above or below it. This option lends itself to a number of interesting effects, such as flying carpets and offset surfaces.

6.3 PASTING TECHNIQUES

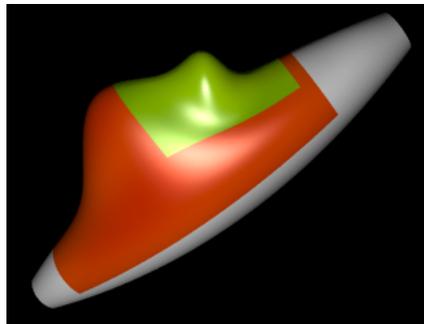
There are three ways to perform the pasting operation: two involve a detail already built, while one uses only the base surface.

I – PARAMETRIC PASTING

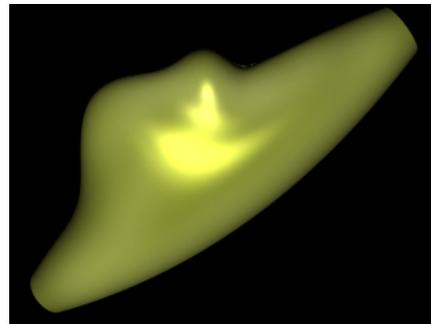
The area on the base surface where the feature will land is defined by four intersecting isoparametric curves – two in the U direction, and two in the V direction. The feature domain is then mapped onto the target rectangle. The feature will stretch and squash on the base surface to cover more or less of it when any of the four defining isoparms slides on the base.

By mapping a flat spline grid over an entire paste hierarchy between (0 and 1 in both U and V) one can simulate skinning or shrink-wrapping. The skin will mold itself onto the hierarchy, reflecting any deformations in the underlying structure because the skin simply becomes the child of the entire structure below it, as in the images shown below.

Using this technique, skinning a set of muscles becomes a matter of setting up the musculature as a pasted composition, followed by pasting a flat grid onto it parametrically. As one or more muscles flex, those pasted on them will react, and the cumulated deformation will propagate naturally to the skin.



Hierarchy (composition)

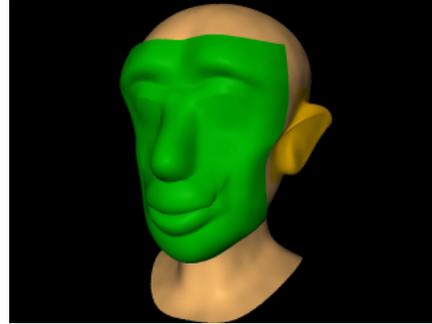


Final skin obtained by pasting a single grid over the entire composition

Parametric mapping also provides a means for turning the hierarchy into a single spline surface. As the muscle example above illustrates, an entire hierarchy becomes enveloped in one, outer-skin surface. Similarly, you can place a smooth mask over an entire composition as show in the facial skin example below.



Pasted feature surfaces



Grid mask pasted on face

The mask is simply a flat grid, pasted parametrically and offset slightly over the head. It is important to note that changes to facial muscles will be propagated to the mask.

In addition, Houdini's conversion operator (Convert SOP) offers the ability to turn an entire paste hierarchy into a single polygonal mesh that preserves all the visual attributes of the composition, including color, transparency, and texture. The fitting operator can later convert the mesh into a spline surface of arbitrary degree.

2 – PROJECTED PASTING

If you would rather position the feature close to the base surface, then project it onto the base as a pasted detail, you would employ projected pasting. Choose as many features as you need, orient them in space relative to the base, set the projection direction or ask for the vectors of minimum distance between feature and base, and you are done.

The features will land on any part of the hierarchy that the projection takes them to. Houdini can accommodate overlapping features and does not mind if the pasting area is not rectangular or if it is not aligned parametrically with the base domain.

3 – SPAWNING

If you want to grow detail from the surface rather than paste one already built, you can spawn a new feature by using the Paste operator. Define the area you want to extract from the base by using the same four isoparms encountered in parametric pasting. However, instead of mapping an external surface onto that region of the base, Houdini peels that section off the base surface and turns it into an instant feature. Since the spawned feature surface is a carbon-copy of its parent (sharing even its parameterization) full mathematical continuity is guaranteed.

The feature can then be refined and modeled, and the spawning process can continue recursively with another and another sub-patch. There is no limit to the number of layers a paste hierarchy can have. Furthermore, other surface and paste hierarchies can be pasted onto it using the projective and parametric techniques.

Spawning lends itself well to the construction of offset surfaces, which are often encountered in industrial design. To that end, spawn a feature as large as the base surface by selecting the end isoparms, then offset it positively or negatively to give it thickness. For small displacements along surface normals, spawning can simulate perfect or near-perfect thickness with excellent results.

6.4 TEXTURING

Texturing is pasting's second nature. Because of its dual nature, a pasted composition can be textured as a single surface, or as a collection of surfaces.

Spawned features inherit all the attributes of their base, including texture coordinates, which means that the mathematical continuity of the surface extends naturally to its texture attributes as well. A spawned composition will yield perfect texture distribution, maintaining continuity across all layers.

Any projected texture, (e.g. orthographic), computes a continuous texture map over the paste hierarchy. This is similar to painting the hierarchy, where the visible layers touched by the brush absorb the texture in an intuitive fashion. In fact, painting a hierarchy is exactly the same as painting a cluster of independent surfaces. The paint program can treat the features as stand-alone surfaces, without having to know anything about paste hierarchies.

Ultimately, pasting can enforce a continuous texture space regardless of the texture coverage of its component surfaces by embedding the texture coordinates in the domain mapping. In other words, by mapping not only the domain of the feature, but also its texture coordinates, pasting is able to bring independent, fragmented texture realms into a unified and continuous space.

6.5 ANIMATION

The inherent parenting between the surfaces captured in the paste hierarchy makes it very easy to produce organic, soft-body deformations and special effects that would otherwise be impossible or prohibitively difficult to accomplish.

Unlike traditional structures, which require that all vertices be captured for IK animation, pasting allows animators to capture and animate only the root surface of the hierarchies. This is possible because everything pasted on the root moves and deforms with it as a result of the hierarchical nature of pasting. Furthermore, animating the base or several of the lower layers is akin to operating on a low resolution model during the set up stages, knowing that all the work will be transferred automatically to the high resolution model in the final stage.

The animation of the pasted features is in no way restricted. They can be animated as independent surfaces, with their motion or deformation added to that of their bases. It is thus possible to generate secondary and tertiary animation with both precision and ease.

For example, facial animation can benefit tremendously from the pasting paradigm: start by animating the underlying facial sets – jaw, brow, cheeks – and watch how the pasted features behave. If they need to be more expressive or must act in peculiar ways, animate them too. The great advantage is that most of the work is taken care of by the base animation, so features might require only minor touch-ups.

As long as the feature boundaries are not explicitly displaced, the detail will remain glued to the underlying surfaces.

6.6 CONCLUSIONS

Pasting is more than a surface refinement instrument for modeling. It provides the ability to render a multi-resolution surface selectively, layer by layer. Pasting simplifies the animation process by allowing you to focus on the behaviour of the base surface since all the pasted features deform with it. Also, base surfaces can act as sliding paths for pasted clusters whose shape and motion are determined by the underlying topology, generating organic effects.

Pasting can be taken further by operating not only on the shape of surfaces, but on their visual attributes as well, such as texture, color, and transparency.

6.7 NOTES

- Paste uses the isoparametric structure of the surfaces to determine how they fit together. In order to get as even a structure as possible it is a good idea to place a Basis SOP, with Parametrization set to *Chord Length*, just before the Paste on the feature surface input.
- Use the *Belt Width* and *Divisions* values in the Paste OP to fine tune the seam if necessary. The Belt settings can be used to increase the number of isoparms in the seam region and can give a better result.
- When using the parametric method of pasting, it's usually best to make the featured surface close in size and proportions to its dimensions after pasting.

GENERAL RULE

If you're going to be animating the surface that you paste onto another surface (like lips or eyelids), put them as close to the bottom of the SOP chain as possible. Pasted stuff that won't animate, like ears and nose, should be higher up. That way, you minimize cooking of Paste SOPs, which is slow.

ANOTHER GENERAL RULE

Your underlying mesh needs to be dense enough so that you can distort parts of it without other parts going along for the ride. For example, a nose changes volume because there are too few points around the nose in the underlying surface. As points are pulled, the nose goes along for the ride.

- Use a Basis OP before pasting a surface.
- Set the parameterization to *Chord Length* or *Uniform*.

BELT WIDTH CONSIDERATIONS

The belts increase the number of isoparms around the edge of the pasted surface. If the belt width is zero, the number of belt divisions doesn't matter. One solution is to put an *if* statement in the *Belt Width* fields that says:

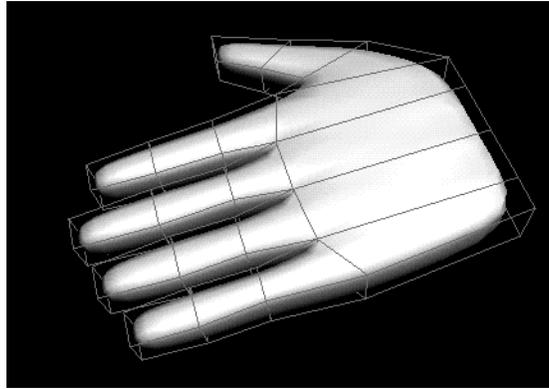
```
if($REZ, 0.1, 0)
```

This means that if my \$REZ variable is set to 1, then you get a belt that's 0.1 in width. Otherwise, the belt is zero. By using this variable for all Paste SOPs, you can turn the belts on and off with one change. The belts slow down playback when animating (as there's more detail in each paste). This is one way of trying to speed things up a bit.

MORE INFO

- For more information on Pasting, see the Paste SOP in: *Ref > Geometry*, and the *Pasting* section in *Geometry Types* section.
- Cristin Barghiel is a senior modeling programmer at Side Effects Software. He holds a Masters of Mathematics from the University of Waterloo, where his thesis on Pasting was supervised by Dr. Richard Bartels.

7 SUBDIVISION SURFACES



7.1 INTRODUCTION

Subdivision surfaces are a middle road between the usual modelling choices of polygons and Splines. On the one hand, polygons are fast and flexible but don't lend themselves to making smooth surfaces. Alternatively, Splines are very good at smooth curves and surfaces but are restricted to rectangular patches which means that complex shapes need a lot of joints and trims, and branching shapes like hands are difficult to build. Subdivision surfaces gives the user the power to create a smooth surface but based on a polygon control shape. As the control shape changes so does the smooth shape within. Another benefit of subdivision surfaces is that they allow the modeller to add extra local detail as needed while maintaining smoothness but without adding a lot of extra geometry to the entire object.

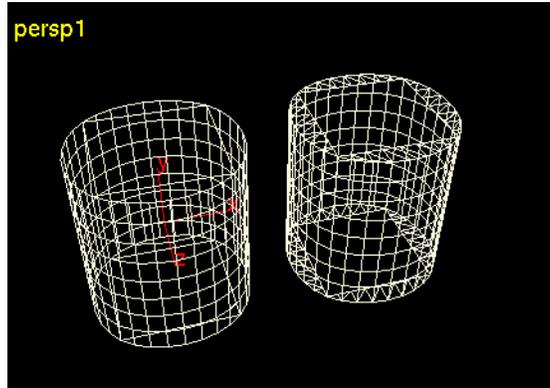
7.2 SUBDIVISION OP

In Houdini, subdivision surfaces are implemented with the Subdivision Surfaces OP. This works like any other operation so that subdivision surfaces can be used and manipulated just like other geometry types. It's possible, for example, to create a series of subdivision surface conversions each on a different part of the same model and each with its own level of detail and characteristics.

The Subdivision Surfaces OP operates just like other OPs. It takes an input polygon, which can be piped into one or both inputs, and divides each face to create a smoothed polygon surface. The parameters control the Groups used, the Depth or level of detail of the resulting surface and the effect of Creases.

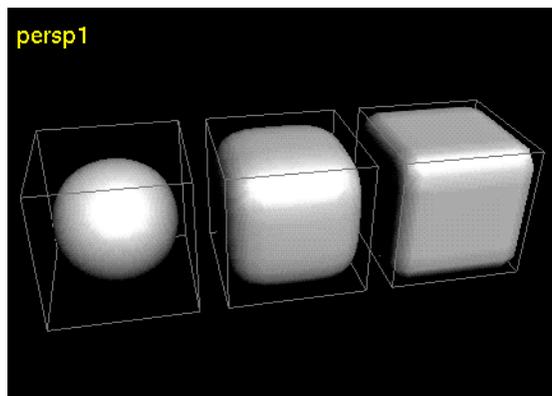
GROUPS

Rather than use the entire polygon to generate the surface it is possible to specify a group of surfaces to effect. By entering the poly numbers for four sides of the box in the *Group* field the Subdivide SOP generates a tube shape as shown. All the same parameters apply except that a few additional ones now take effect. Notice that the end faces of the tube remain squared off. To close these and generate a single shape, you can either pull the hole closed or stitch them using the relevant controls in the parameters area.



7.3 CREASES

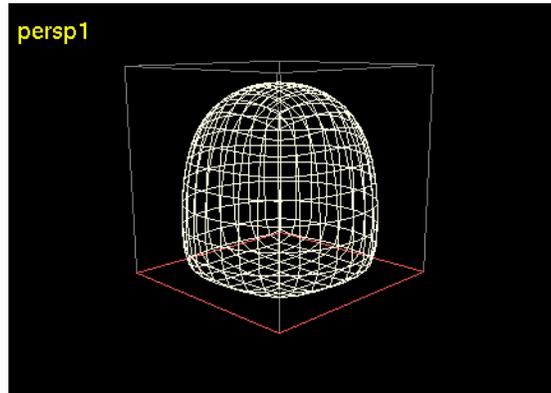
Creases are a powerful tool for manipulating the shape and smoothness of subdivided surfaces. They work by controlling the strength of the pull of the polygon faces on the subdivision surfaces, like a magnet drawing the surface towards the reference polygon. The figure below shows the result of setting the *Crease Weight* to 0, 1, 2 reading from left to right. As the weight increases so the pull effect strengthens and the shape approaches the reference polygon.



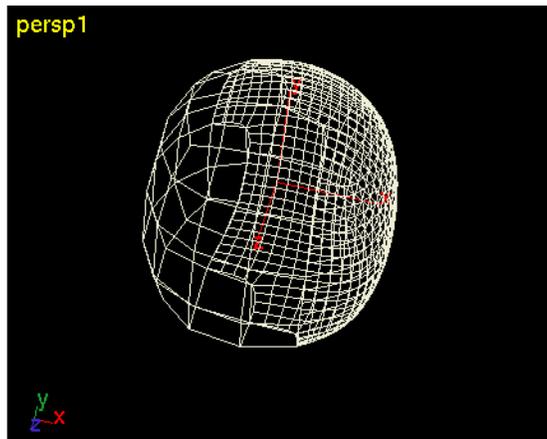
Note: The Crease attribute applied by the Crease Operation will be used directly by the Photorealistic Renderman renderer, but will not be used by the Mantra renderer due to patent issues.

APPLYING CREASES

Creases can be applied selectively using the *Creases* field which operates just like the *Group* field just above it (or with the *Crease OP* p. 524). The example below shows the result of putting the base face in the *Creases* field and entering a non-zero value for the *Crease Weight*. Note that if you do not have the vertex attribute set then you will need to wire up the right hand Subdivide SOP input to the preceding SOP, which in this case is the Facet SOP.



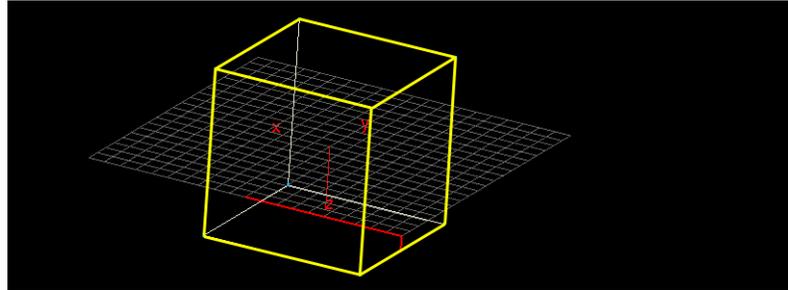
It is also possible to re-subdivide a surface or part of a surface. The following example shows the result of applying a Subdivide SOP to part of the object by selecting and transferring a selection to the subdivision surfaces *Group* field. You can use this if you need to add local detail but still want to maintain smoothness. The entire surface will still conform to the basic polygon and to any deformations or transformations you apply taking the extra detail along with it.



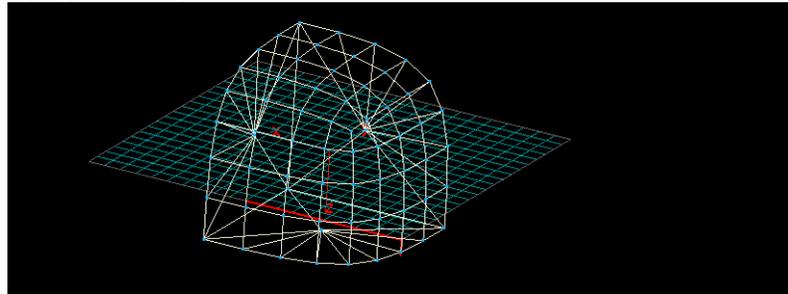
7.4 MODELLING SUBDIVISIONS

Subdivision can be applied in a fast interactive fashion using the Subdivision OP. It is possible to select either faces, with the left mouse button, or edges, with the middle mouse button, and subdivide them by right mouse clicking.

The illustration shows the Subdivision Surfaces State enabled with three faces of the cube selected.



By right clicking, the system subdivides the three faces as shown.



The default state for the Subdivision Surfaces SOP parameters is to have the *Override Crease Weight Attribute* enabled. So you can simply set a value which applies to all the creases. You can, however, set a crease attribute using the Vertex or Primitive SOPs which allows for more control.

2 Geometry Operations

I ADD OP

I.1 DESCRIPTION

This OP can both create new Points and Polygons on its own, or it can be used to add Points and Polygons to an existing input.

If an input is specified, this OP adds points and polygons to it as specified below. If no input is specified, then it generates the points and polygons below as a new entity.

I.2 PARAMETERS – POINTS PAGE

POINT [0-5]

The three input fields represent the X, Y and Z coordinates of the point. The last input field represents the spline weight of the point. If the point is later used to create a spline (NURBS or Bezier) primitive, the weight will influence the shape of the primitive and may cause that primitive to become rational. Polygons and metaballs are not affected by this weight. These values can be constants (numbers) or variables. Below are three examples:

- 1: `0.2 0.42 1.3`
- 2: `0.2 ch("../xform1/tx") 1.36`
- 3: `point("../grid1",5,"P",0), point("../grid1",5,"P",1), point("../grid1",5,"P",2)`

In the third example, we are reading the sixth point (0...5), from the OP, *grid1*.

DELETE GEOMETRY BUT KEEP THE POINTS

Use this option to remove any unused points. When checked, existing geometry in the input are discarded, but the polygons created by this OP are kept, as well as any points in the input.

I.3 PARAMETERS – POLYGONS PAGE (BY PATTERN)

POLYGON [1-6]

Create a fixed number of polygons by specifying a point pattern for each polygon. Enter *connection lists* here to add polygons. These consist of a list of point numbers to define the order in which the points are to be connected.

The form is: {from}-{to}[:{every}][,{of}] .

examples of valid connection lists

1 2 3 4	Makes a polygon by connecting point numbers 1,2,3,4.
1 3-15 16 8	All points from 3-15 are included.
1-234 820-410 235-409	Points from 1-820 are included, in the specified order.
0-15:2	Every other point from 0 to 15 is included.
0-15:2,3	Every 2 of 3 points are included (i.e. 0, 1, 3, 4, 6, 7, 9, 10, 12, 13, 15).
!4	Every point except 4 is included.
!100-200	Every point <100 and >200 is included.
*	Include all points.
9-0	The first ten points are included in reverse order.
!9-0	All but the first ten points are included in reverse order.

closed polygons

To create a closed a polygon, check the *Closed* button beside it.

REMOVE UNUSED POINTS

Keep only the connected points, and discard unused points.

I.4 PARAMETERS – POLYGONS PAGE (BY GROUP)

Create as many polygons as determined by the group field and by the grouping / skipping rules.

GROUP

Subset of points to be connected (accepts *Pattern Matching* p. 38).

ADD

Optionally join subgroups of points:

<i>All Points</i>	Adds all points just as if you added them manually in the <i>Points</i> page.
<i>Group of N Points</i>	Adds only the number of points specified.
<i>Skip every Nth Point</i>	Adds points, but skips every Nth one.
<i>Each Group Separately</i>	Creates separate polygons for each group specified in the <i>Group</i> parameter. For example, if you have a Group OP creating a group called <i>group1</i> and using the <i>Create Boundary Groups</i> option, you can connect this to an Add OP and enter <i>group1__*</i> in the <i>Group</i> parameter. If <i>Each Group Separately</i> is chosen, polygons will be created for each boundary on the surface.

Tip: The *Each Group Separately* option is useful when pasting surfaces. Boundary groups can be created for the boundaries of two adjacent surfaces, and then the PolyLoft OP (using the *Points* option) can be used to stitch these surfaces together.

N */inc*

Increment / skip amount to use for adding points.

CLOSED

Closes the generated polygons.

1.5 USES

Used in conjunction with a point expression, the Add OP can be useful for extracting a specific point from another OP. For example, to extract the X, Y and Z value of the fifth point, from a Grid OP in *geo1*:

```
point("geo1/grid1",5,"P",0),
point("geo2/grid1",5,"P",1),
point("geo3/grid1",5,"P",2)
```

Points added in this way are appended to the end of the point list if a Source is specified. Click the Information Pop-up on the OP Tile to find out how many points there are. For example, if you have added two points and there are 347 points (from 0 to 346), you have added the last two point numbers: 345 and 346.

I.6 LOCAL VARIABLES

N	The \$N variable returns the index of the last point in the geometry including the points created in this OP.
PT0 - PT5	These variables represent the point index numbers added in the Point page. This is useful because you can refer to the points in the connection list with: <i>\$PT0-\$PT5</i> rather than using absolute index numbers, which change with every new input source.
CEX, CEY, CEZ	Defines the centroid of the input geometry.
X-Y-ZMIN, X-Y-ZMAX	Defines the extent of the bounding box for the input geometry.
SIZEX, SIZEY, SIZEZ	The size of the bounding box of the input geometry.

I.7 SEE ALSO

- *Group OP* p. 592
- *Point OP* p. 667
- *PolyLoft OP* p. 679
- *Carve OP* p. 486 > *Extract Points* parameter

2 ALIGN OP

2.1 DESCRIPTION

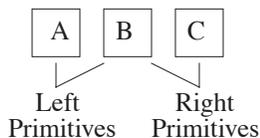
This OP aligns a group of primitives to each other or to an auxiliary input, by translating or rotating each primitive along any pivot point.

LEFT AND RIGHT PRIMITIVES

The notions of “left” and “right” which follow depend on context. If an auxiliary input is used, it is always the right primitive and the primary input geometry are all left primitives. If only one input is used, then for each pair being aligned, there is a left and a right primitive. This means that relative to neighbouring primitives, one primitive can be both left and right.

Case 1

Three primitives in the first input, no secondary input.

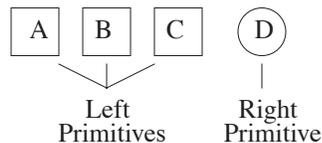


Result

A will be aligned to B
B will be aligned to C

Case 2

Three primitives in the first input, a secondary input containing a sphere.



Result

A will be aligned to D
B will be aligned to D
C will be aligned to D

2.2 PARAMETERS

GROUP

A subset of primitives to align (accepts patterns, as described in: *Scripting > Pattern Matching* p. 38). If blank, it aligns the entire input.

ALIGN

Can optionally align subgroups of n primitives or every n th primitive in a cyclical manner. Example: Assume there are six primitives numbered for 0 - 5, and $N = 2$. Then:

- Groups will generate 0-1 2-3 4-5
- Skipping will generate 0-2-6 and 1-3-5.

N */inc*

Determines the number of primitives to be either grouped or skipped. $N \geq 2$.

2.3 PARAMETERS – ALIGN PAGE

BIAS */bias*

Determines which primitive remains unaffected:

0	Left
1	Right

LEFT UV */leftuv1 /leftuv2*

Pivot Location for each “left” primitive.

RIGHT UV */rightuv1 /rightuv2*

Pivot location for each “right” primitive.

RIGHT UV END */rightuvend1 /rightuvend2*

If an auxiliary input is used, this location specifies an end point for the alignment.

Left primitives are then distributed uniformly between the Right UV and the Right UV End.

INDIVIDUAL ALIGNMENT

Causes each primitive of the input to be aligned. If unchecked, only the first primitive is aligned and all others are placed relative to it, preserving the spatial layout of the left primitives.

TRANSLATE

When enabled, translates primitives during alignment by translating the left UV position to the right UV position.

ROTATE

When enabled, rotates primitives during alignment by aligning the left UV tangents (at the left UV position) to the right UV tangents (at the right UV position).

2.4 PARAMETERS – TRANSFORM PAGE

TRANSLATE / ROTATE ORDER

Sets the overall transform and rotation order for the transformations. The transform and rotation order determines the order in which transformations take place. Depending on the order, you can achieve different results using the exact same values.

TRANSLATE / ROTATION / SCALE / PIVOT /tx-y-z /rx-y-z /sx-y-z

Allows you to perform a post-alignment transformation. Specify the amount of translation / rotation / scaling about the local XYZ axes.

2.5 USES / WORKS IN RELATION WITH

Works with all primitive types.

2.6 INPUTS / GEOMETRY TYPES**ALIGN SOURCE**

Any geometry type. Treated as left/right primitive pairs or left primitives if an auxiliary source is used.

AUXILIARY SOURCE

Any geometry type consisting of a single primitive treated as the “right” primitive.

2.7 SEE ALSO

- *Carve OP* p. 486
- *Creep OP* p. 525
- *Rails OP* p. 713
- *Sweep OP* p. 791

3 ARM OP

3.1 DESCRIPTION

The Arm OP creates all the necessary geometry for an arm, and provides a smooth, untwisted skin that connects the arm to the body. It is controlled through inverse kinematics linked to a handprint.

3.2 PARAMETERS – ARM PAGE

CAPTURE TYPE

You can use either *Ellipses* or *Capture Regions* as deformation geometry. Ellipses are for use with the Skeleton OP. Capture Regions are for use with the Capture OP.

ARM AXIS

Position the model along the +X or -X axis.

RADIUS */bonerad*

Controls the scale of the circle radii.

ROTATE HAND */rotatehand*

This parameter rotates the hand and the wrist joint to match the orientation of the hand-print object. In order to operate correctly, the end-effector (hand print) scale transformations must remain at 1.

Note: If the channel is set to 0, then the hand rotations are relative to the forearm. If the channel is set to 1, the hand rotations are the same orientation as the end effector.

AUTO ELBOW TWIST */autoelbow*

This parameter affects the default twist of the elbow joint to a more natural position.

ELBOW TWIST */elbowtwist*

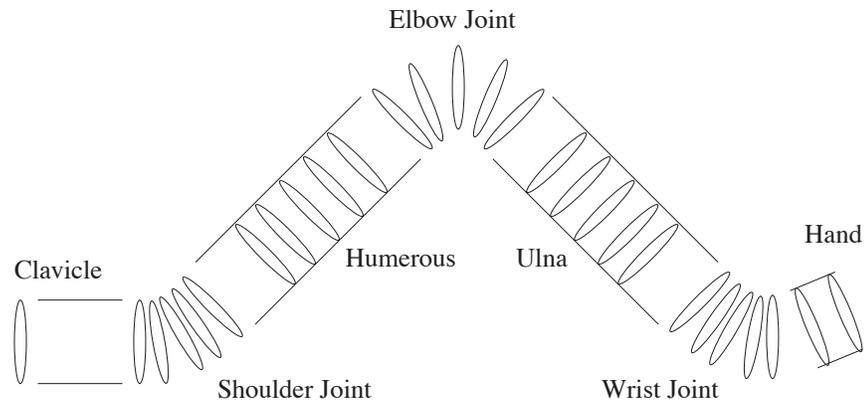
Specifies the rotation angle of the elbow joint.

FLIP ELBOW

This toggle positions the arm using an alternative elbow position solution.

3.3 PARAMETERS – LENGTHS PAGE

ARM LENGTHS



The following parameters control bone lengths, as illustrated above.

- Clavicle */clavlength*
- Humerus */humlength*
- Ulna */ulnlength*
- Hand */handlength*

These parameters control the joint lengths, as illustrated above:

- Shoulder Joint */shoulder*
- Elbow Joint */elbow*
- Wrist Joint */wrist*

3.4 PARAMETERS – TRANSFORMS PAGE

When the arm is positioned to reach the end affector (hand print), the shoulder, elbow and wrist joints may produce unnatural looking bends. The transform fields allow manual adjustment of each controlling circle of each joint to fix this.

Each joint circle (e.g. Shoulder 1) is given three transform fields (two translates and one scale). These values are scaled by the amount of bend applied to the particular joint. In other words, when the arm is fully extended, the transforms have no effect. When the arm joint angles are at 90°, they have maximum effect. Thus, set the joints to 90° before adjusting these values.

3.5 PARAMETERS – END AFFECTOR PAGE

AFFECTOR OBJECT

Allows the end affector to be another object, or simply defined by a default box, which is controlled by the transformations below.

TRANSLATE / ROTATE / SCALE */tx /ty /tz /rx /ry /rz /sx /sy /sz*

These values apply a transformation on the end affector (handprint) to reorient and reposition it. For a full explanation of transforms, see the *Transform OP* - p. 805.

3.6 USES / WORKS IN RELATION WITH

The arm can be used as input geometry to the Skeleton OP, or it to create actual arm geometry. To create usable geometry, append a Convert OP and a Skin OP to the arm.

TIP

If you are used to working with polygon circles in *action*, and still desire this type of functionality, you can convert them to primitive circles (necessary for the Skeleton, Arm, and Limb OPs) using the Convert OP.

3.7 SEE ALSO

- *Capture OP* p. 472
- *Convert OP* p. 512
- *Limb OP* p. 616
- *Skeleton OP* p. 745
- *Skin OP* p. 749

4 ATTRIBUTE OP

4.1 DESCRIPTION

The Attribute OP allows you to manually rename and delete point and primitive attributes. The RenderMan page remaps attributes especially for the creation of .rib data streams for use with the RenderMan renderer.

4.2 PARAMETERS – POINT PAGE

RENAMING ATTRIBUTES

The top portion of this page deals with the renaming of attributes. Specify the name of the original incoming attribute in the first column, and the new name you want it renamed to in the second column.

DELETE ATTRIBUTE

At the bottom of the *Point* page, there is a field to specify attributes to delete. Simply enter a list of the attributes (separated by spaces) to delete. For example entering:

```
Cd Alpha
```

Will delete point colours and point alpha from the geometry. The names of these attributes can be determined by viewing them in a OP's info pop-up.

pattern matching

Deletion accepts general pattern matching when determining which attributes to delete. For example:

*	= Delete all attributes.
Cd	= Delete the Cd attribute.
Cd Alpha	= Delete Cd & Alpha attributes.
a*	= Delete all attributes beginning with <i>a</i> .
* ^Cd	= Delete all attributes except Cd.

Note: You should never delete just one of the following attributes, but always keep/delete them together. This is because Houdini's capture/deform system expects these three attributes to occur together. Deleting a subset of the three will cause errors as they are interdependent:

```
pCapt (point attribute)
pCaptPath
pCaptData (detail attribute)
```

4.3 PARAMETERS – VERTEX / PRIMITIVE / DETAIL PAGES

These tabbed pages are identical to the *Point* page except that they deal with vertex and primitive attributes of the input geometry.

4.4 PARAMETERS – RENDERMAN PAGE

This tabbed page allows you to precisely specify the mapping of Houdini attributes to RenderMan attributes. When Houdini saves the Geometry for the RIB stream, these attributes will be declared properly in the RIB stream so that RenderMan shaders can pick them up correctly.

The four columns in the RenderMan page specify: the Houdini attribute name, the RenderMan attribute name, the RenderMan attribute type and an optional offset into the Houdini attribute.

ADD DEFAULT MAPPINGS

Enable this option if there are no specific overrides necessary when saving to a RIB stream. The defaults are as follows:

Attribute	Variable	Type	Offset Value
Cd	Cs	Vertex Color	0
Alpha	Os	Varying Color	0
N	N	Varying Vector	0
uv	s	Varying Float	0
uv	t	Varying Float	1
rest	Pr	Vertex Point	0

Note the use of the offset field in the “uv” to “t” mapping. This specifies that the second float of the “uv” attribute will be used as the “t” coordinate for RenderMan.

Warning: Some combinations of “Vertex” type attributes will cause prman to core dump. For example, if both “Cs” and “Os” are mapped as “Vertex Color”, then prman will core.

RENDERMAN ATTRIBUTE TYPES

uniform

This is the equivalent of a Houdini Primitive attribute, in other words, the attribute is uniform over the whole primitive.

constant

This RenderMan data type is similar to the Uniform type and should, if possible, be used instead of the Uniform type. For a detailed explanation of the difference between uniform and constant data types, please see the Pixar Application Note #22 (which is shipped with the prman documentation).

vertex

Requires prman version 3.6. This attribute type is specifically for NURBS surfaces so that internal vertices can have an effect.

varying

This attribute type works well for every surface other than NURBS. For example, with polygons, or a mesh, the attribute is interpolated between vertices of the primitive. However, for NURBS, only the four corner attributes are used.

MAPPINGS

float

A single floating point value, for example “particle id” or “red color”.

color

Houdini assumes that colors are three channels (i.e. RGB) For example, “Cs”.

point

A point in space specified by three floating point values. e.g. “Pr” (rest position).

vector

A vector attribute is transformed differently than a point attribute. The vector represents a direction. For example, “N” (normal) or “v” (velocity).

normal

This attribute (a floating point triplet) defines the geometry’s normal. By default, Houdini outputs normals as the “normal” attribute type instead of “vector” in order to support Blue Moon Rendering Tools and other RIB renderers.

EXAMPLES

Take the third component (offset 2) of the color attribute and map it to the RenderMan attribute called “BlueColor”.

```
"Cd" "BlueColor" "vertex float" 2
```

Take the color attribute and assign it to be uniform across the primitive:

```
"Cd" "Cs" "uniform color" 0
```

To use these attributes in a shader, you must declare the attributes at the end of the shader parameter list. Then they can be used within the shader as any other variables. For example:

```
surface myShader(float myparm, vertex float BlueColor) { ... }
```

4.5 INPUTS / GEOMETRY TYPES

- Accepts geometry of all types for remapping, renaming, and deletion.

4.6 USES / WORKS IN RELATION WITH

- Other OPs destined to be rendered using RenderMan.

5 ATTRIBCOPY OP

5.1 DESCRIPTION

AttribCopy is used to copy attributes between groups of vertices, points, or primitives.

Attributes are copied from a source group to a destination group. The source and destination group must be of the same type, but there is no restriction on the attribute class.

When copying, a general rule is applied when the size of the destination is greater than the size of the source. In this case, the elements of the source are repeated in a cyclic fashion. An analogous example follows:

Given a source pattern "ABC" and a destination pattern of length 8, the resulting destination pattern is "ABCABCAB".

The same logic applies to vertices, points, and primitives. The order in which the source and destination groups are specified will affect the result of the copy.

If the copied attribute does not exist on the destination geometry, it is created automatically.

EXAMPLE

Use the AttribCopy Operation to copy UV texture coordinates from one face to another. The group type should be primitives, and the attribute class should be points or vertices (depending on where the UV attribute has been applied).

5.2 PARAMETERS

SOURCE GROUP

A subset of the source geometry from which to read attribute values.

GROUP TYPE (SOURCE)

The type of elements referenced in the Source Group field.

DESTINATION GROUP

A subset of the destination geometry whose attribute values will be overwritten by the values contained in the source geometry.

GROUP TYPE (DESTINATION)

The type of elements referenced in the Destination Group. The Source and Destination Group Types must be identical.

ATTRIBUTE

The attribute to copy. "Color" and "Texture UV" are provided for convenience. Other attributes can be specified by name by selecting "Other Attribute".

ATTRIBUTE NAME

When "Other Attribute" is selected, this is the name of the attribute to copy.

ATTRIBUTE CLASS

Where the attribute is located on the geometry. Vertex, point, and primitive attributes can be copied. "Use Group Type" looks for the attribute in the same place as specified by the *Group Type* parameter. "Auto Detect" first looks in the place specified by the Group Type, then looks for the attribute in vertices, points, and primitives (in that order).

5.3 SEE ALSO

- *Attribute OP* p. 436
- *AttribMirror OP* p. 444
- *Point OP* p. 667
- *Primitive OP* p. 697
- *Vertex OP* p. 852

6 ATTRIBUTE CREATE OP

6.1 DESCRIPTION

You can make a point, vertex, primitive, or detail attribute. You can specify default and initial values. The initial value is specified with all the appropriate local variables.

You can also specify a variable name to have this attribute be referred to by in future SOPs.

EXAMPLES

- Adding the vector attribute "abc" will create local variables \$ABCX, \$ABCY and \$ABCZ.
- Adding the float attribute "abc" of size 2 will create the local variables \$ABC1 and \$ABC2.
- If the size is 1, the variable \$ABC will be created, as will \$ABC1.

6.2 PARAMETERS

NAME

Name of the attribute

LOCAL VARIABLE

Name of the local variable

CLASS

Where to add the attribute to the geometry.
Can be a point, detail, primitive, or vertex attribute.

TYPE

Float, Integer, Vector, or String.

SIZE */size*

Number of elements in the attribute.

DEFAULT */default*

Default attribute value

VALUE */value*

Value to set attribute to.

STRING */string*

Value to set the string to.

LOCAL VARIABLES

The standard local variables are usable in this Operation.

See *Local Variables* p. 671.

TECHNICAL NOTE

If you are using these with the HDK, it is important to note that strings are added as "Index" attributes, not as actual "String" attributes.

6.3 SEE ALSO

- *Attribute OP* p. 436
- *Point OP* p. 667
- *Primitive OP* p. 697
- *Vertex OP* p. 852

7 ATTRIBMIRROR OP

7.1 DESCRIPTION

AttribMirror mirrors attributes from one side of a mirror plane to another.

A correspondence is established between the geometry on the source side of the model (the side in which the arrow is on) and the closest mirrored geometry on the destination side.

The closeness of geometry is defined by position. A primitive's position is computed as the center of its bounding box. Vertex attributes are mirrored by first finding a mirror primitive, then finding a mirror point within that primitive. The result is that vertex attributes can be mirrored perfectly in a symmetric model.

The mirror plane itself is considered to be on the destination side of the mirror. To correspond two geometry elements, one must be on the source side and the other must be on the destination side.

EXAMPLE

Use the texturing tools to texture one side of a symmetric model (such as a human-like character) and then use AttribMirror to automatically mirror the UV texture coordinates to the opposite side of the model.

7.2 PARAMETERS

GROUP

A subset of the geometry to use as a source or destination for the mirror.

GROUP TYPE

The type of elements referenced in the Group field.

USE GROUP AS

As a source, the elements of the group are mirrored onto the rest of the geometry across the mirror plane. As a destination, the elements of the rest of the geometry are mirrored onto the elements of the group across the mirror plane.

ATTRIBUTE

Attribute to mirror. "Color" and "Texture UV" are provided for convenience. Other attributes can be specified by name by selecting "Other Attribute".

ATTRIBUTE NAME

When "Other Attribute" is selected, this is the name of the attribute to mirror. The attribute must exist on the geometry elements specified by the Group Type parameter.

TOLERANCE

Create a correspondence between mirrored geometry only if the mirrored distance lies within this tolerance.

ORIGIN */origin{x,y,z}*

The origin of the reflection plane.

DISTANCE */dist*

Distance to move reflection plane.

DIRECTION */dir{x,y,z}*

Direction of the reflection plane's normal.

7.3 SEE ALSO

- *Attribute OP* p. 436
- *AttribCopy OP* p. 440
- *Point OP* p. 667
- *Primitive OP* p. 697
- *Vertex OP* p. 852

8 ATTRIBPROMOTE SOP

8.1 DESCRIPTION

AttribPromote is used to promote and demote attributes from one type to another. This provides a straightforward way to convert an attribute from one class to another. For example, a point attribute can be converted to a primitive attribute, using any one of a number of different merge methods.

For index attributes, the merge methods treat the attribute as a string. Average will perform median, and the other numeric methods will just resort to First Match.

For attributes with multiple components (such as vector), each merge is done on each component independently.

8.2 PARAMETERS

ORIGINAL NAME

This is the name of the attribute in the source class.
E.g.: "Cd" will pick colour.

ORIGINAL CLASS

This is the class which the attribute starts in.
It can be one of point, primitive, detail, or vertex.

NEW CLASS

This is the class to which the attribute should be promoted. It must be different than Original Class or a warning is raised and nothing is done.

PROMOTION METHOD

Whenever there is more than one attribute in the original class which matches a single entity in the new class, some method must be used to determine the new value. For example, if converting from point attribute to primitive attribute, there will be many points corresponding to a single polygon.

<i>Maximum</i>	Picks the largest match.
<i>Minimum</i>	Picks the smallest match.
<i>Average</i>	The mean, or the sum of all matches divided by the number of matches.
<i>Mode</i>	The most common match. If there is more than one most common, it will be the minimum of the most common.

<i>Median</i>	The middlemost of the matches. If there is an even number of elements, it is the higher of the two possible middles.
<i>Sum</i>	All of the matches added together.
<i>Sum of Squares</i>	All of the matches squared, and then added together.
<i>Root Mean Square</i>	The square root of the average of the squares of all the matches.
<i>First Match</i>	The first valid match. For promoting points to primitives, this would be the point of the first vertex in the primitive.
<i>Last Match</i>	The last valid match. For promoting points to primitives, this would be the point of the last vertex in the primitive.

CHANGE NEW NAME

If not set, the new attribute name will match the original name.

NEW NAME

The new name for the attribute if Change New Name is not set.

DELETE ORIGINAL

If set, the original is deleted after promotion.

8.3 SEE ALSO

- *Attribute OP* p. 436
- *Attribute Create OP* p. 442

9 BAKE VEX OP

9.1 DESCRIPTION

BakeVEX pre-shades mesh, Bezier or NURBs geometry with the appropriate surface and displacement VEX shaders. The lights in the scene are used to compute illumination on the geometry. A camera must be specified to determine specular information.

- No shadows, reflections or refractions are computed.
- This Operation only works with mesh, Bezier or NURBs surfaces.
- The results of the shaders are stuffed into the geometry's attribute data.

9.2 PARAMETERS

CAMERA

The viewing camera for shading.

BAKE SURFACE SHADER

Run the appropriate surface shader.

CF / OF / AF ATTRIBUTE(S)

The Cf / Of / Af variables go into this attribute.

BAKE DISPLACEMENT SHADER

Run the appropriate displacement shader.

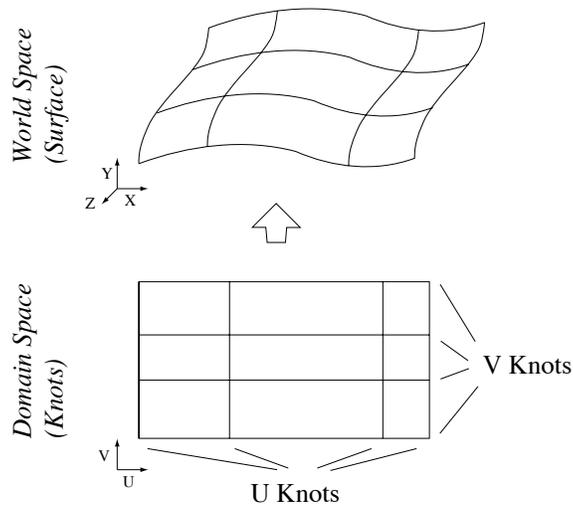
P ATTRIBUTE

The resulting position goes into this attribute.

10 BASIS OP

10.1 DESCRIPTION

This OP provides a set of operations applicable to the parametric space of spline curves and surfaces. The parametric space, also known as the “domain” of a NURBS or Bézier primitive, is defined by one basis in the U direction and, if the primitive is a surface, another basis in the V direction. The size of the domain is given by the values of the knots that make up the basis.



The Basis OP contains both ratio-preserving and non ratio-preserving operations.

If the basis reparameterization does not change the distance ratios between knots, the shape of a NURBS primitive is not affected. If the ratios are not preserved, however, a NURBS primitive will change shape in the area influenced by the modified knots; furthermore, if the primitive is a NURBS or Bézier surface, any profiles it may contain will be affected as well.

For more information about bases and knots see *Breakpoints, Knots, and Spline Basis* p. 239 in the *Geometry Types* section.

10.2 PARAMETERS

GROUP

Group of spline primitives (accepts patterns, as described in: *Scripting > Pattern Matching* p. 38). Non-spline types are ignored.

Two sets of pages follow, one for each parametric direction (U and V). In each set the operations are applied in tab order, starting from the left: *Parameterization*, *Mapping*, and then raising the spline *Order*. To disable all the operations of a set, toggle off the U or V check mark above it. The V set is meaningful only for spline

surfaces, and is ignored otherwise. Channel names are given for both the U and V pages.

10.3 PARAMETERS – PARAMETERIZE PAGE

These operations on this page are not shape-preserving.

PARAMETERIZATION

<i>Unchanged</i>	Does not change the basis.
<i>Uniform</i>	Distributes all the knots evenly, maintaining the basis origin and length. Recommended only for regular shapes.
<i>Chord length</i>	Computes the knot ratios based on the distances between successive CVs. This is the most common and most effective parameterization method.
<i>Centripetal</i>	Similar to the chord length method. Recommended for shapes containing sharp turns.
<i>Manual:Single</i>	Loads the basis with the knots listed in the “Knot Sequence” below. If the OP input contains several primitives, only the basis of the first spline primitive will be affected.
<i>Manual:Propagated</i>	Same as above + it copies the basis to the basis of all the other spline primitives in the model or in the group. Using this feature on curves that have the point count and degree will generate cleaner surfaces, i.e. surfaces with fewer ioparms.
<i>Knotslide</i>	Shift clusters of knots within the basis. See <i>Knotslide (From Parameterization Menu)</i> p. 451 for discussion.

READ BASIS

Loads the original knots of the basis into the “Knot Sequence” field when the *Parameterization* type is *Manual*.

KNOT SEQUENCE (UNLABELED)

The basis of the first spline primitive in the input loads its knot sequence with the data specified in this field when the *Parameterization* is set to *Manual*. The values must be in ascending order, and their total count must match the number of knots in the basis. To ensure an exact count, click on the “Read Basis” button to read the original knot sequence into this field.

Note: Bézier bases cannot have repeated knots. NURBS bases accept repeated knots as long as the knot multiplicity does not exceed the degree of the basis. The first two and last two knots of a NURBS basis must be identical.

KNOTSLIDE (FROM PARAMETERIZATION MENU)

Allows individual knots or clusters of knots to be shifted within the space of the basis to which they belong. This provides a more interactive method than *Manual* reparameterization. When *Knotslide* is enabled in the *Parameterization* menu, two fields become available:

range */urange1 /urange2*

Range specifies the domain interval to be shifted. All the knots captured in this range are shifted by the same amount as far as the closest neighbouring knot on either side.

bias */ubias*

Bias indicates the direction and the amount of translation. A *Bias* of 0.5 does not displace the knots at all. As the *Bias* decreases, the knot cluster migrates closer to its left-neighbouring knot. A *Bias* greater than 0.5 forces a migration to the right.

Sometimes a *Bias* of 0 or 1 does not clamp the knot cluster to the closest neighbouring knot. The reason for this behaviour is that the knot multiplicity cannot be allowed to exceed the *degree* of the basis. For example, an order 4 (degree 3, or “cubic”) spline can have at most 3 identical knots in sequence.

When sliding the knot of a NURBS basis, a larger area of that spline is affected. This may create the impression that more knots than those in the cluster are being displaced.

10.4 PARAMETERS – MAP PAGE

The operations contained on this page are shape-preserving.

CONCATENATE

Indicates whether the bases of the input spline primitives should be concatenated such that the last knot of the first primitive coincides with the first knot of the second primitive, and so on. This operation is performed before the ones below it, thus allowing a whole set of bases to be mapped onto a given interval (usually [0,1]) while enforcing basis continuity.

ORIGIN */uorigin /vorigin*

The new origin of the basis, or the origin of the cumulated bases if concatenation is *On*.

LENGTH*/ulength /vlength*

The new length of the basis, or the total length of the cumulated bases if concatenation is *On*. The length, which represents the distance between the first and last knot, must be greater than zero.

SCALE*/uscale /vscale*

The multiplier applied to the basis starting at the basis origin. The scale must be greater than zero.

10.5 PARAMETERS – ORDER PAGE**RAISE TO***/orderu /orderv*

The only operation here is raising the order (or degree) of the spline basis. Valid orders range from 2 to 11. Orders lower than the current spline order are ignored. The operation preserves the shape of the primitive.

Production Tip: Before applying a spline-based texture projection with the Texture OP, remap the U and/or V bases of the spline surface between 0 and 1 to ensure a complete mapping of the texture. If a single texture map must be shared by several surfaces, the surface bases should be concatenated prior to being remapped.

10.6 SEE ALSO

- *Refine OP* p. 719
- *Resample OP* p. 723
- *UV Texture OP* p. 843

11 BLEND OP

11.1 DESCRIPTION

This OP provides 3D metamorphosis between shapes with the same topology. It can blend between sixteen input OPs using the average weight of each input's respective channel. It will also interpolate point colors and / or texture co-ordinates between shapes.

For best results, there should be the same number of points / CVs (and faces) in the geometry of each OP. The points should have a similar order; if point 17 in one source is on the left side, and point 17 in another Source is on the right side, it will move across the object during the blend, which may create distorted and twisted shapes. To ensure that this doesn't happen, you can make all the pieces to be blended by editing point positions of one common base shape. Each of the pieces to be morphed must be in different OPs.

11.2 PARAMETERS

GROUP

Specifies a point or primitive group in the first input. If, for example, a group is specified containing the first and third points, then the first and third point of every input will be blended whereas the second, fourth, fifth, etc. points will be set to match the first input source. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

BLEND 1-4 ... 13-16 */blend1 ... /blend16*

Channels controlling the contribution of the geometry inputs to the output geometry.

DIFFERENCING

Generates exaggerated blends between objects where values above 1 or less than 0 will result in over-scaled blends.

When this option is checked, the above channel values are not summed and scaled to 1. The first input is considered a reference; Blend computes the difference between the first input and the others for each point; values greater than 1 and less than 0 produce exaggerations of the shapes for inputs 2 and higher. The first input, however, cannot be exaggerated by its blend channel, */blend1*, and it is considered to be the "base". When using Differencing, the */blend1* channel has no effect. If the geometry in the first input must be deformed, feed it into another input, where the blend channels have an effect.

BLEND POSITION

When checked, the point positions of the inputs will be blended based on the weights of the blend channels. If not checked, the input geometry will not change, only allowing Blending of Colors, Normals and Textures if selected.

BLEND COLORS

When checked, the point colors of the geometry inputs will be blended based on the weights of the blend channels.

BLEND NORMALS

When checked, the point normals of the geometry inputs will be blended based on the weights of the blend channels.

BLEND TEXTURE

When checked, the point texture co-ordinates of the geometry inputs will be blended based on the weights of the blend channels.

INPUT EDITOR

Adjust the order of the geometry inputs. Up to sixteen input OPS can be managed.

11.3 INPUTS / GEOMETRY TYPES

Accepts all geometry types for up to sixteen inputs.

11.4 USES / WORKS IN RELATION WITH

This OP can be used as a form of geometry deformation. Model your geometry, such as the mouth on the head of a character. This will become your first input.

11.5 EXAMPLE

Set the input to the first OP to be blended. For example, if there are five shapes to be blended, the simplest set-up is the following:

1. Use four Curve OPs, and modify each copy from the base geometry to be deformed / morphed.
2. Append a Blend OP off of the OP containing the base geometry and then proceed to connect up the four OPs into the Blend OP.
3. The amount that each input contributes to the resulting shape is controlled by each input's blend channel. Editing the values of the blend channels will result in each input to determine the final shape based on ratios.

For example, if the */blend3* channel's value is 1, and the rest of the channels are set to 0, the resulting shape is the geometry in the third input OP. Now change the value in the */blend5* channel to 1, the resulting shape will be a 50/50 percent blend of the geometry contained in the input OPs' three and five. You would get the exact same result by setting both channels to 0.3, or to any number if the numbers are the same in both channels, there will be equal contribution to the shape.

12 BOOLEAN OP

12.1 DESCRIPTION

The Boolean OP takes two closed polygonal sets, A and B. Set these Sources to the OPs with the 3D shapes that you wish to operate on.

Note: The Boolean OP handles polygonal geometry. For boolean-type operations with NURBS and Bezier surfaces – see *Cookie OP* p. 516 and *Surfsect OP* p. 788.

This OP is quite visual and intuitive; you can experiment with the different combinations on screen to see the effects. There are two important requirements for input geometry:

- Shapes must be completely closed. A tube with open ends is not an acceptable input. You can close Tube ends with an end-cap. An extruded letter with no back polygons output is also unacceptable (even with backs output it can be unacceptable because of the second requirement).
- All polygons must be convex and coplanar. In the case of the Extrude OP, you must select *Output Convex Faces* for front and back faces. It will now be a usable input for Boolean. The Divide OP also offers a convexing function for geometry not created with the Extrude OP.

Other caveats for *Boolean* are the following:

- Point colors and texture UV coordinates are not interpolated correctly.
- In some cases polygons can be reversed so that all normals point outwards. The polygon reversal should not usually present much of a problem (use a *Primitive OP > Face/Hull page > Vertex > Reverse* to reverse them if necessary).

12.2 PARAMETERS

OPERATION

Some of the operations below produce guide geometry to give you visual feedback on the results of the operation being performed. The appearance of the geometry is context sensitive—if you are performing an intersect operation, or either of the edge operations the guide will be both inputs; if you are doing A minus B then the guide will be B and if B minus A then the guide will be A. If you are doing union then there will be no guide geometry.

If the guide geometry is too distracting, you can disable it by entering the Viewport options dialog and clicking on the Guide geometry button so that it no longer appears indented. This procedure is global and will disable the guide geometry of other OPs as well.

Select one of the following operations from the pop-up menu:

<i>Union</i>	Geometry from the two inputs is combined, and interior polygons are removed. The result is a closed shape. This can be very useful for joining pipes or other shapes that the camera will travel inside, or where the intersecting shapes must be transparent. Points at the intersection of the shapes are not consolidated.
<i>Intersect</i>	The resulting geometry is a closed shape where the two input shapes overlap or intersect. Geometry outside the common area is discarded.
<i>A minus B</i>	The result is a closed shape in which the geometry from B is cut away or subtracted from the geometry in A.
<i>B minus A</i>	All operations same as previous three, but resulting shape is Part B with Part A cut away from it.
<i>A Edge / B Edge</i>	Closed face(s) are produced, at the edges where the two parts meet. The face(s) can be twisted (not planar). The point order is unpredictable; append a Polygon OP with the <i>Order Points</i> option enabled to sort them out if you want to make connections to the face.

The Boolean OP will automatically orient polygons so they face the same way. This may not be enough in some cases because *Boolean* results in some unshared edges where the intersection cut took place. If the shading is still not good enough, you are best to follow *Boolean* with a Facet OP. In it, *Consolidate Points*, *Orient Polygons* and finally *Cusp*.

If you have really strange shaped polygons, you can first triangulate one or both of the inputs with the Divide OP.

ACCURATE ATTRIBUTE INTERPOLATION

If selected, all inputs are convexed to triangles, otherwise they are convexed to quadrilaterals.

CREATE GROUPS

If selected, a group is created containing all faces pertaining to the first input, and a second group containing all faces of the second input.

Group A	name of first group to create
Group B	name of second group to create

12.3 INPUTS / GEOMETRY TYPES

Accepts only polygonal inputs.

12.4 USES / WORKS IN RELATION WITH

- Cutting a 3D shape from another 3D shape.
- Joining a 3D shape to another 3D shape, removing interior polygons.
- Creating a 3D shape where two 3D shapes intersect.
- Creating holes or planes where 3D shapes intersect.
- Animating any of the above effects.

12.5 SEE ALSO

- *Cookie OP* p. 516
- *Curvesect OP* p. 537
- *Surfsect OP* p. 788

13 BOUND OP

13.1 DESCRIPTION

Bound computes and creates a bounding volume for the input geometry. The bounding volume can be either a box or a sphere. The bounding box can be used in the Lattice operation.

13.2 PARAMETERS

GROUP

Subset of geometry to bound.

GROUP TYPE

The type of elements referenced in the Group field.

KEEP ORIGINAL

Keep the input geometry.

BOX SUB-PAGE

Compute bounding box.

divisions */divsx /divsy /divsz*

Number of divisions along xyz axes.

enforcement bars

Places diagonal crossbars in the divisions of the box.

minimum size

The smallest size you will allow the bounding box to be.

SPHERE SUB-PAGE

Compute bounding sphere.

orientation

The poles of the sphere will align itself with the axis specified.

accurate bounds

Use a more accurate (but slower) bounding sphere calculation.

minumum radius

The smallest sizze you will allow the bounding sphere to be.

13.3 SEE ALSO

- *Box OP* p. 461
- *Lattice OP* p. 609

14 BOX OP

14.1 DESCRIPTION

Creates cuboids which can be used as geometries by themselves, or can be subdivided for use in the *Lattice OP* p. 609.

If it has an input then it will create a box that bounds the incoming geometry. Otherwise, the parameters determine the size and location of the box.

14.2 PARAMETERS

PRIMITIVE TYPE

The faces of the generated box geometry can be of several types: Mesh, NURBS or Bezier surfaces. You can also specify connectivity, number of rows, columns, Uorder and V order.

CONNECTIVITY

<i>Rows</i>	Creates horizontal lines.
<i>Columns</i>	Creates vertical lines.
<i>Rows & Cols</i>	Both Rows and Columns.
<i>Triangles</i>	Build each face of the box with triangles.
<i>Quads</i>	Build each face of the box with four-sided quadrilaterals.

SIZE */sizex /sizey /sizez*

Size of the Box or Cube along the X, Y, and Z axes.

CENTER */tx /ty /tz*

These X, Y, and Z Values determine where the center of the Box is located.

AXIS DIVISIONS */divrate*

Divides the box along each axis into the number of divisions specified. Boxes divided in this way do not appear when rendered because the divisions consist of open polygons.

AXIS ORDERS */orderrate*

The order of the surface on each axis.

DIVISIONS*/divsx /divsy /divsz*

The number of divisions in X, Y, and Z to split this Box into.

ENFORCEMENT BARS

Places four diagonal crossbars in each division of the Box.

CONSOLIDATE CORNER POINTS

Ensures that the box's corner points are shared.

14.3 INPUTS / GEOMETRY TYPES**BOUNDING SOURCE**

If an input is used, a bounding box of its geometry is made, and the options *Center* and *Size* are disabled.

14.4 USES / WORKS IN RELATION WITH

You can use the points as an input geometry for the Lattice and Spring OPs.

14.5 SEE ALSO

- *Bound OP* p. 459
- *Lattice OP* p. 609
- *Spring OP* p. 769

15 BRIDGE OP

15.1 DESCRIPTION

This OP is useful for skinning trimmed surfaces, holes, creating highly controllable joins between arms and body, branches or tube intersections.

The Bridge OP is similar to the Skin OP but with much greater control over the resulting surface. Given a set of profiles (i.e. curves on surface) and/or spatial faces, the Bridge OP builds a NURBS skin with specified tangent and curvature characteristics. The precision of the resulting surface is highly dependent on the number of required cross-sections and on the quality of the profile extraction. High precisions will generate a very dense surface with, potentially, many multiple knots.

In general, the higher the order of the curve, the better the fit the Bridge OP will be able to provide. However, it is generally better to stick to cubics (order 4) curves, as the software is optimized for cubics.

Because the Bridge OP can join both a set of spatial curves and trim curves, it can be used much like the Skin OP and/or the Fillet OP. However, bridging trimmed surfaces is more expensive than bridging carved surfaces.

You will usually need a Trim, Bridge, or Profile OP after a Project OP.

- Use a Trim OP to cut a hole in the projected surface
- Use a Bridge OP to skin the profile curve to another profile curve.
- Use a Profile OP to extract the curve on surface or remap it's position.

Note: To texture-map the resulting skin, use an Orthographic projection rather than a Spline-based projection. This results in better continuity across the surfaces.

15.2 PARAMETERS

GROUP

The *Group* edit field allows you to enter profile groups for profiles and/or faces to bridge. This is optional if you have regular geometric curves or surfaces, however, you must enter something here in order for Bridge to work with profile curves. For example *.0 will Bridge the 0th (first) profiles of all incoming primitives.

Note: Always specify the curves on surface if you want the Bridge OP bridge curves on surfaces; otherwise it will attempt to bridge free-floating curves.

BRIDGE

Allows bridging of subgroups of N primitives or patterns of primitives.

N */inc*

Determines the pattern of primitives to bridge using this OP.

ORDER */order*

Sets the spline order for both profile extraction and skinning operations.

15.3 PARAMETERS – SURFACE PROPERTIES PAGE

MIN. X-SECTIONS */isodivs*

The minimum number of cross-sections in the resulting skin. If you create a high-density surface, Houdini's level of detail may display the surface less smoothly than it actually is. You can increase the level of detail by adjusting the viewdisplay options (e.g. *viewdisplay -l 1.5 sopmain.persp1*) for the Viewport.

Production Tip: If, in generating a smooth surface, you create an extremely complex surface, some of the complexity can be removed without damaging the appearance of the surface by appending a Refine OP, and using its *Unrefine* option. In the Refine OP, set the *First U* parameter to zero and, in the *Unrefine* option's parameters, set the U value close to the order of the surface created in the Bridge OP.

USE

Specifies the type of normal to use for computing direction:

- | | |
|---------------------|---|
| <i>Frenet Frame</i> | The Frenet Frame of the face. This option uses a local coordinate system on the curve to compute the direction. |
| <i>Normal</i> | The Normal of the face. |

CIRCULAR ARC FILLETS

Tells Houdini to try to generate a round fillet rather than a free-form fillet. Only the sign (positive or negative) of the tangent scales is used; the scale magnitude is ignored when building a circular fillet.

The radius of the fillet is computed automatically and adjusted according to the distance between the rails (curves and/or profiles) and their tangents.

ROTATE / SCALE TANGENTS */rotatet1-3 /scalet1-3*

The scaling and rotation parameters contain three fields:

- the first field applies to the first face in the input.
- the second field applies to all intermediate faces.
- the third field applies to the last face in the input.

The rotation fields (degrees) apply further rotation to the tangents, while the scale parameter further scales the tangents.

USE CURVATURE

Takes curvature into consideration as well.

SCALE CURVATURES */scale1 - 3*

Further scaling of the curvature.

Note: If the resulting skin bulges too greatly, you can achieve a smooth resulting transition between surfaces by disabling the *Preserve Tangent & Preserve Curvature Magnitude* parameters, and manually tweaking the Tangent Scales and the Curvature Scales. In general, avoid tweaking the Rotations of the Tangents unless you wish to deform the resulting surface.

If the bridge bulges on one side but not the other, try increasing the *Min. Number of Cross sections* in the bridge.

15.4 PARAMETERS – PROFILE EXTRACTION PAGE

This page's parameters are similar to those found in the Fit and Project OPs

DIVISIONS PER SPAN */sdivs*

Number of 2-D points evaluated in each span.

TOLERANCE */tolerance*

Precision of 2-D fitting algorithm.

PRESERVE SHARP CORNERS

Enables or disables fitting of sharp turns. If cracks appear in the resulting skin, *Preserve Sharp Corners* is usually a good solution; however, it may add additional knots which can create undesirable “ripples” in some cases.

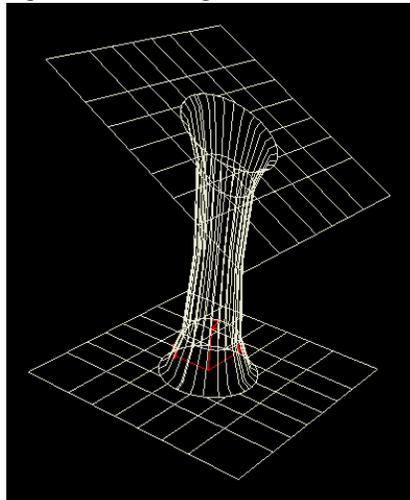
If this option is disabled, fewer isoparms are generated and the surface may not follow the contours of the profile curves perfectly unless the profile curves were built using the *Preserve Sharp Corners* option.

15.5 EXAMPLE

JOINING TWO TRIM (PROFILE) CURVES

1. Place a Circle OP. Primitive Type: NURBS; Radius = 0.2, 0.2
2. Place a Grid OP. Primitive Type: NURBS.
3. Feed both the Circle and Grid OPs into a Project OP. Make it the display OP. You notice the projected circle on the grid - our trim curve.

4. Append a Trim OP and make it the display OP. Turn on Gouraud shading for the Viewport – you now see the trimmed holes in the surface of the grid.
5. Append a Copy OP. Number of Copies: 2; Translation Z: 1.0; Rotation X: 30. Make it the display OP. Now we have two grids with trimmed holes in them.
6. Append a Bridge OP, and make it the display OP. Scale Tangents: 0, 0, 0; Use Curvature: On; Preserve Curvature Magnitudes: Off; Scale Curvatures: 3, 3, 3 . Nothing happens. Why?
7. We need to specify which profile curves to skin. Turn on *Profile Numbers* in the Viewport options (click  at the bottom-right of Viewport, and enable the icon). We can see the profile numbers of the two trim curves are 0.0 and 1.0 – meaning the 0th profile of the 0th primitive and the 0th profile of the 1st primitive). The strange numbering is because primitive numbers start at 0 instead of 1.
8. In the Bridge OP's Group field enter: *.0 – this means to include the 0th (first) curve from all (the * wildcard character) primitives in the skin. You now see the resulting bridge between the two trim curves. The skin bulges outwards.
9. We can control the bulge by playing with the *Scale Curvatures* and the *Tangent Magnitudes*. Set the Scale Curvatures to: -3, -3, -3 . Now we have an inward-bulging tube connecting the two holes.



10. Experiment with moving the location and size of the holes (change the Translation and Radius in the Circle OP). The Bridge OP dynamically updates the geometry connecting the two surfaces. Setting the Scale Curvature to: 0, 0, 0 produces a straight-through connection between the two holes.

15.6 SEE ALSO

- *Fillet OP* p. 574
- *Project OP* p. 707
- *Skin OP* p. 749
- *Trim OP* p. 819

16 BULGE OP

16.1 DESCRIPTION

Bulge deforms the points in the first input using one or more magnets from the second input. The "magnetic field" of influence is defined by a metaball field. This operation is a simpler version of the Magnet operation.

16.2 PARAMETERS

DEFORM GROUP

Points to be deformed with the magnet.

MAGNET GROUP

Magnet primitives in the 2nd input.

MAGNITUDE */mag*

Scaling factor for point displacement.

NORMALIZE WEIGHT

Normalize the displacement vector from the metaball origin to the point.

16.3 SEE ALSO

- *Magnet OP* p. 632
- *Metaball OP* p. 638

17 CAP OP

17.1 DESCRIPTION

Caps are used to close open areas with flat or rounded coverings. Meshes are capped by extending the mesh in either the U or V direction (e.g. a NURBS tube). Faces are capped by appending a separate face or hull cap.

17.2 PARAMETERS – U CAPS / V CAPS

PRESERVE NURB SHAPE U / V

When capping a NURBS surface, use this option to preserve the original surface by clamping it at the point of capping.

FIRST U / V CAP

Select an option from the menu:

<i>No End Cap</i>	Leave this side of the primitive unaffected.
<i>End Cap Faceted</i>	Cap by creating a face of a similar type which uses unique points.
<i>End Cap Shared</i>	Cap by creating a face of a similar type which shares points with the original primitive.
<i>End Cap Rounded</i>	Cap by creating / extending a mesh containing a number of concentric circular regions. This creates a bullet or domed shaped cap.

DIVISIONS /divsu1 /divsu2 /divsv1 /divsv2

Number of cross sections in the rounded cap.

SCALE /scaleu1 /scaleu2 /scalev1 /scalev2

Affects the height of the rounded cap (both positive and negative).

LAST U / V CAP

Similar to *First Cap U / V*, but builds a cap on the other end of the primitive in the opposite direction.

17.3 SEE ALSO

- *Torus OP* p. 796
- *Tube OP* - p. 822
- *Revolve OP* p. 730

18 CACHE OP

18.1 DESCRIPTION

In essence this OP records its input geometry for faster playback. It should be used when cook times for a chain of OPs is long and a quicker playback is needed.

Once cached, the geometries can be accessed in any order. This is advantageous to a 2D flipbook or scene render since the geometry is still fully 3Dimensional. It is also ideal for POP networks since once cached it can be played forward or backward.

The upshot being that you can scrub otherwise sluggish animations in real time, play POP networks backwards, etc. because things are precomputed and stored in a cache.

18.2 PARAMETERS

START/END/INC */range*

A range of values at which to set the the index and param name when caching.

INDEX */index*

Set to each value within the range for each geometry cached.

INDEX PARAM NAME */param*

A stampable parameter set to the index during caching.

SET FRAME TO INDEX WHEN CACHING */indextime*

Set the local time to the index for each geometry cached.

LOAD */loadmethod*

The loading behaviour: *As Needed*, or *All on Next Cook*.

RELOAD ALL CACHE */reload*

Clear the cache and reload everything.

CLEAR CACHE */clear*

Delete all the stored cache.

CACHE POINTS ONLY */cachepoints*

Store a single topology for the first cached geometry and only point data for the remaining geometries.

BLEND POSITION */blendpos*

Interpolate points between geometries.

18.3 LOCAL VARIABLES

none.

19 CAPTURE OP

19.1 DESCRIPTION

Capture works in conjunction with the Deform operation and the Capture Region operation by assigning capture weights to all points that fall within a Capture Region(s). The closer the point is to the boundary of a Capture Region, the lighter the assigned capture weight to that region will be.

The optional second input can be used to specify extra capture regions to process in addition to those in the Parent Object hierarchy.

The Capture operation does will not visibly change the geometry (except to possibly add point colors for feedback). To actually change the geometry by animating the Capture Regions, append a Deform operation.

Typically one or more Capture Regions are placed in an object (usually a bone) and named "cregion". The object is then read into a Capture operation through the object's hierarchy. The object is moved to deform the Capture operation's geometry with a Deform operation.

The weighting scheme is described in the next section, *Capture Edit OP* p. 476.

Note: When a hip file is loaded, all of the associated capture regions are evaluated at the capture frame. Keyframes must be set to position the capture regions properly at the capture frame. Otherwise, the geometry will be weighted incorrectly upon subsequent file loads.

19.2 PARAMETERS – CAPTURE PAGE

HIERARCHY

An object hierarchy is traversed to find the Capture Regions with which to do the weighting. This parameter specifies the top of the traversal hierarchy.

REGION OP

Specifies how to find the Capture Region OP within the objects in the traversed object hierarchy. Three choices are available: OPs named "cregion" (default), Display OPs and Render OPs.

WEIGHT FROM

Surface (default) uses the surface to compute the weight of a point (or CV in the case of NURBS). This is very helpful for NURBS surfaces when the CV's may be far off the surface. By using Weight from Surface, each CV is weight based on the surface's location within a Capture Region. This is determined by calculating the CVs "most influenced point" on the NURBS surface and computing the weight of that position. Using the Weight from Point choice, the location of the point within the region is used for the weighting.

CAPTURE FRAME

Frame number at which to compute the capture weighting. This frame will cook slower than other frames due to the weighting computation.

Frame number at which to compute the capture weighting. Every time Houdini reaches this frame, the geometry will be re-captured. This frame will cook slower than other frames due to the weighting computation.

It is a common practice to set the capture frame to a frame outside of your animation range, -1 for example.

Note: When a *.hip* file is loaded, all of the associated capture regions are evaluated at the capture frame. Keyframes must be set to position the capture regions properly at the capture frame, otherwise the geometry will be weighted incorrectly upon the subsequent file loads.

NORMALIZE WEIGHTS

Adjusts the capture weights so they add up to one for every point. This makes the paint tools easier to use. For backwards compatability, this should be changed to off.

POINT COLORING

This option colours each point by capture region (using point attributes) according to its capture weight. The points inherit their colours from the Capture Region(s) in which they lie. For example, if a point falls within a blue capture region and also a yellow capture region, the point will be coloured green (more blue near if the blue weighting dominates, and more yellow if the yellow weight dominates). In addition, the points become darker as the weighting gets lightened. For example, near the edge of a blue region, a caught point will appear dark blue. Near the centre, it will appear bright blue. If a point is not caught by any region, it is coloured bright red.

ZERO WEIGHT POINT COLOR

This parameter is used for points with zero weight when doing point colouring. The default zero weight point colour is white.

19.3 PARAMETERS – OVERRIDE PAGE

OVERRIDE FILE

The name of the capture override file (**.ocapt*). this file is loaded after Houdini completes its point weighting. Each line of the override file lists a point number, a region (path and primitive number) and a weight. The points on the geometry are modified to use these custom weights.

override file format (.ocapt)

Each line in the override file must have the following format:

```
point_number region_path region_prim_num weight
```

For example:

```
0 /obj/chain_bone1/cregion 0 0.0
0 /obj/chain_bone2/cregion 0 3.757989
0 /obj/chain_bone3/cregion 0 1.757989
```

weights point 0 to three regions (actually only two because the first entry has a weight of zero). Remember that the override file is read after Houdini does its own capture weight computation, so in this case, if point 0 was originally assigned to region */obj/chain_bone4/cregion 0*, this part of its weighting would be unchanged.

There is no upper limit to the number of regions that can be weighted to a point. If a point/region combination is in the file twice, the second one is used. For example:

```
0 /obj/chain_bone1/cregion 0 1.0
0 /obj/chain_bone1/cregion 0 2.0
```

causes the first entry to be ignored (a weighting of 2.0 is used).

The weight field is used to prescribe the relative amount of influence a region has on a point. It can be any number. The range of the weights computed by Houdini are specified by the *Inner/Outer Weight* parameter of each Capture Region (please see *Capture Edit OP* p. 476).

The easiest way to a capture override file is to use the *Save Override File* button (see below) and start from the file produced by Houdini.

SAVE FILE

The file specified here can be used as a “working file” to save the point weighting of all the points or a selected subset of points. The file format for the capture override files is fairly straight-forward (see above), so this is a good place to start if you need to do custom overriding.

INCREMENT SAVE FILE

This increments the save file name before saving. Be careful about turning this option off because there is no warning or confirm dialog to prevent you from overwriting an *.ocapt* file.

SAVE ALL DATA TO FILE

This saves the point weighting of all points to the *Save File*.

SAVE SELECTED POINTS TO FILE

This saves the point weighting only for the points that are selected in the viewport. Note: you must be editing this particular OP in the Viewport for this selection to apply to this OP.

19.4 SEE ALSO

- *Capture Edit OP* p. 476
- *Deform OP* p. 541
- *Objects > Character Tools* p. 376

20 CAPTURE EDIT OP

20.1 DESCRIPTION

CaptureEdit is used to edit point capture weights. It is designed to be used by the Edit Capture Weight operation at the object level. It maintains a history of capture weight changes by storing within itself a list of capture weight differences from the input geometry.

Typically, its input geometry is a Capture or CaptureProximity operation, and then its output is connected to a Deform operation. Capture or CaptureProximity generates an initial weighting of the points using a list of bone objects. Then, CaptureEdit is used to modify and tweak the resulting capture weight attributes. Finally, a Deform operation is used to deform the geometry based on those capture weight attributes.

20.2 PARAMETERS

GROUP

The points to modify.

CREGION

A string composed of the path to the capture region to set the weight. For example, "chain_bone1/cregion 0" specifies use of primitive 0 in the operation named cregion in the chain_bone1 object. The primitive number within the geometry must match a primitive generated by the CaptureRegion operation.

WEIGHT

The weight of the capture region to set to in all the points.

START NEW WEIGHT

Starts a new capture weight modification

UPDATE POINT COLORS

Adds diffuse point colors to the geometry to show the effect of the modified capture weights.

ZERO WEIGHT POINT COLOR

Specifies the color to use for zero weighted points when "Update point colors" is on.

EXTRA CAPTURE FRAME

Specifies the capture frame to use when modifying weights of capture regions that have not already been captured in the input geometry.

20.3 SEE ALSO

- *Capture OP* p. 472
- *Capture Proximity OP* p. 484
- *Capture Paint OP* p. 480
- *Deform OP* p. 541

21 CAPTURE MIRROR OP

21.1 DESCRIPTION

Capture Mirror mirrors capture attributes from one half of a symmetrical model to another. A correspondence is established between each point of the source side of the model (the side in which the arrow is on) with the closest mirrored point on the destination side. This saves having to re-apply their touchups to both halves, since there is no other way to mirror capture weights.

The From and To fields specify a list of capture regions that are to be mirrored. Both the from and to list orders are important. For example, to mirror the right arm to the left arm one might have:

From: "ruparm/cregion 0" "rloarm/cregion 0" "rhand/cregion 0"

To: "luparm/cregion 0" "lloarm/cregion 0" "lhand/cregion 0"

21.2 PARAMETERS

GROUP

Specifies a subset of points to affect.

USE GROUP AS

Specifies whether the Group field specifies the set of source or destination points.

BEHAVIOUR

How to treat unspecified capture regions. If it is "Copy", the destination point's captures weights are replaced by the source point's capture weights. Then, each of the "from" bones is changed to a "to" bone and vice-versa.

If it is "Leave", the capture weights of unspecified capture regions will be unaffected. Capture weights of any "from" bones in the source points will result in identical capture weights for corresponding "to" bones in the destination points.

ORIGIN */origin{x,y,z}*

The origin of the reflection plane.

DISTANCE */dist*

Distance to move reflection plane.

DIRECTION $/dir\{x,y,z\}$

Direction of the reflection plane's normal. This normal should face the source points when "Copying" unspecified capture regions.

FROM

This is a space delimited list of /obj relative paths of capture regions to act as the source side bones.

TO

This is the same as From, but specifies the destination bones. The two lists form a one-to-one correspondence so their order is important.

21.3 SEE ALSO

- *Capture OP* p. 472

22 CAPTURE PAINT OP

22.1 DESCRIPTION

Capture Paint is a specialized paint operation for painting capture attributes.

Note: Refer to Interface > Brush Tools p. 139 for useful info!

22.2 PARAMETERS – OPERATION PAGE

OPERATION

Paint is to paint onto the surface. Eye dropper mode will read the attribute from the surface into the fore/background colour.

MERGE MODE

How the color is to be applied to the surface.

ACCUMULATE TO STENCIL

By default, the operation is applied and the stencil cleared after every brush stroke. With this set, the stencil is not cleared until "Apply and Clear Stencil" is pressed.

APPLY & CLEAR STENCIL

Applies the operation and clears the stencil. This is equivalent to right-clicking in the viewport.

FOREGROUND/BACKGROUND */fg /bg*

Weights to apply. If "Accumulate To Stencil" is on, the left mouse does foreground and the middle mouse erases.

APPLY TO ALL

Deforms all of the selected geometry.

RESET ALL CHANGES

Restores geometry to initial state.

VISUALIZE ATTRIBUTE

Whether to visualize the weight and range to use.

VISUALIZE

The type of visualization to use.

VISUALIZE MODE

For single bone, how to colour the weights.

ZERO WEIGHT COLOR

The colour of points with no weighting.

NORMALIZE WEIGHTS

Automatically adjust weights so they always add to one.

22.3 PARAMETERS – BRUSH PAGE

SHAPE

The basic shape of the brush: circle, square, or bitmap.

BITMAP

What bitmap to use. The alpha channel becomes the brush.

RADIUS */radius*

The radius of the brush.

BRUSH ANGLE */brushangle*

How far to rotate the brush.

BRUSH SQUASH */squash*

Amount to squash the brush in the y direction before rotation.

OPACITY */opacity*

The amount to affect the stencil mask.

BRUSH SPLATTER */brushsplatter*

A random noise in the brush's opacity based upon the position on the brush.

PAPER GRAIN */papergrain*

A random noise on the object's stencil mask based on the object position.

SOFT EDGE */softedge*

Percentage of the brush to be rolled off.

KERNEL FUNCTION

Which metaball kernel to use for the roll off.

UP VECTOR TYPE

How the brush should be oriented on the surface:

Stroke Direction Oriented in the direction in which the brush moves.

Fixed Oriented as specified in the Up Vector field.

UP VECTOR */upvector*

The fixed up vector to orient brush to.

22.4 PARAMETERS – STROKE PAGE

Note: Be sure to see: Interface > Secrets of the Brush Page p. 142 for useful info!

ORIENT BRUSH TO SURFACE

Switches between the brush being perpendicular to the surface or always oriented along the view direction. If you are having trouble with a shaky brush, try turning this off.

REALTIME MODE

Enable this to paint geometry as it's deforming.

The last few parameters on this page are read-only parameters. While it is theoretically possible to manually apply brush strokes by adjusting these parameters, that would be excruciatingly difficult. Their primary use is to provide feedback as to where the brush is currently hitting your geometry. This can be used as a sort of 'geometry eyedropper' when you want to quickly investigate some troublesome geometry.

DIRECTION

This is the normal on the hit primitive at the hit position.

HIT LOCATION

This is the position in space where the brush ray hit the geometry.

HIT PRIMITIVE

This is the primitive number that was hit by the brush ray.

HIT UV

These are the parametric coordinates where the primitive was hit. They are the UV coordinates used to evaluate the surface with a prim call, for example, and is not to be confused with any "uv" texture coordinates that may be present.

EVENT

This is used internally to track the current state of the drawing mode. No-op is the default and is used to prevent any accidental writes using out of date or irrelevant hit information.

22.5 SEE ALSO

- *Interface > Brush Tools p. 139.*
- *Objects > Paint Capture Weights p. 338*
- *Comb OP p. 504*
- *Sculpt OP p. 735*
- *Paint OP p. 649*

23 CAPTURE PROXIMITY OP

23.1 DESCRIPTION

CaptureProximity works in conjunction with the Deform operation and the CaptureRegion operation by assigning capture weights to points based on their distance to the Capture Regions. The closer the point is to the centre line of the Capture Region, the stronger the weight will be assigned.

The optional second input can be used to specify extra capture regions to process in addition to those in the Object hierarchy.

This operation does will not visibly change the geometry (except to possibly add point colors for feedback). To actually change the geometry by animating the Capture Regions, append a Deform operation.

Typically one or more Capture Regions are placed in an object (usually via bones) and named "cregion". The object is then read into a Proximity Capture through the object's hierarchy. The object is then deformed using the Deform operation in conjunction with the output from this operation.

Note: When a .hip file is loaded, all of the associated capture regions are evaluated at the capture frame. Keyframes must be set to position the capture regions properly at the capture frame. Otherwise, the geometry will be weighted incorrectly upon subsequent file loads.

23.2 PARAMETERS – PROXIMITY CAPTURE PAGE

GROUP

Point group to capture

HIERARCHY

Top of the object hierarchy traversed for the Capture Region operations

CAPTURE FRAME

Frame number at which to compute the capture weighting. This frame will cook slower than other frames due to the weighting computation.

COOK AT

Specifies how often this operation should cook

REGION SOP

Specifies the geometry in which to find the Capture Regions

FORCE RECAPTURE

Forces the operation to cook immediately

23.3 PARAMETERS – WEIGHTING PAGE

WEIGHTING METHOD

Specifies whether capture regions captured by a point need to be connected by their objects

WEIGHT FROM

Weight from either the point location or surface location (for NURBS or Bezier surfaces)

DROP OFF

Specifies the strength of the capturing. Larger values will usually mean that more points are captured.

MAX. INFLUENCES

Specifies the maximum number of capture regions that any point will be captured to

NORMALIZE WEIGHTS

Adjusts the capture weights so they add up to one for every point. This makes the paint tools easier to use. For backwards compatibility, this should be changed to off.

23.4 PARAMETERS – COLORING PAGE

POINT COLORING

Adds point color attributes to show the weighting of the points. The color of each point will be a blend of the colors of its captured regions.

ZERO WEIGHT POINT COLOR

Color used for points with zero weight

23.5 SEE ALSO

- *Capture OP* p. 472
- *Capture Edit OP* p. 476
- *Capture Paint OP* p. 480
- *Deform OP* p. 541

24 CARVE OP

24.1 DESCRIPTION

The Carve OP works with any face or surface type, be that polygon, Bézier, or NURBS. It can be used to slice a primitive, cut it into multiple sections, or extract points or cross-sections from it.

Like the Project OP, it also creates profile curves, but they are extracted as iso-parametric (2D) profiles directly from a surface, whereas the Project OP extracts a 3D curve projected onto a surface.

Note: When using a Carve OP on a trimmed surface, you can't fillet or join the trim curves.

24.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

FIRST U /domainu1

This specifies a starting location to begin the cut/extract. Select this and a parametric U location between 0 and 1.

SECOND U /domainu2

This specifies an ending location to complete the cut / extract. Select this and a parametric U location between 0 and 1.

U DIVISIONS /divsu

This specifies the number of cuts / extracts to be performed between first U and the second U.

FIRST V /domainv1

This specifies a starting location to begin the cut/extract. Select this and a parametric V location between 0 and 1. Applies only to surfaces.

SECOND V /domainv2

This specifies an ending location to complete the cut / extract. Select this and a parametric V location between 0 and 1. Applies only to surfaces.

V DIVISIONS */divsv*

Specifies the number of cuts/extracts to be performed between first V and second V.

CUT

If enabled, it will split the primitives at the locations specified above, creating an array of distinct smaller primitives.

Keep Inside Keep the primitives specified between the first and second locations.

Keep Outside Keep the primitives lying outside the first and second locations.

EXTRACT

If enabled, it will extract a cross-section along each location specified above. Note, if a face is used, only points can be extracted and the V parameters have no effect.

extract 3-d isoparametric curve(s)

The extraction operation creates a 3-D, free-floating curve that matches the surface perfectly at the given U or V location.

extract point(s)

Available for Mesh and surfaces only. If selected, it will create a point at every U, V location specified, rather than removing a U cross section *and* a V cross section.

extract 2-d isoparametric profile(s)

Creates a 2D curve at the specified U/V location that matches the surface perfectly.

KEEP ORIGINAL

If selected, it will not remove the original primitive.

ONLY AT BREAKPOINTS

Performs cutting or extractions at breakpoints for curves and surfaces, at vertices for polygons, and along isoparms for meshes.

By default the *Only At Breakpoints* option will only cut the outer breakpoints within the specified interval. By enabling *Only At Breakpoints*, the U and V divisions parameters are replaced with *Cut At All Internal <U/V Breakpoints*, which cut the primitives at all breakpoints falling within the specified interval when enabled.

24.3 TIPS

OPENING A CLOSED PRIMITIVE

Given a closed surface, it can be converted to an identical open surface at any location by using the Carve OP, and cutting at a single location (e.g. First U enabled, Second U disabled). The same is true for faces.

DICING A FACE OR SURFACE

Select the *Cut* operation, First U = 0, Second U = 1, First V = 0, Second V = 1, with (at least) three divisions for U and V. This will dice the surface into (at least) 3×3 components. This can later be used with the Primitive OP to “explode” the surface.

CUTTING A HOLE IN A SURFACE

Select the *Cut* operation, First U = 0.33, Second U = 0.66, First V = 0.33, Second V = 0.66, 2 divisions for U and V. Keep Inside is off, Keep Outside is on. This will cut a rectangular region from the centre of the surface.

ANIMATING A POINT OVER A FACE OR SURFACE

Select the *Extract* option, First U = x, Last U disabled, First V = y, Last V disabled, Extract Point is on, Keep Primitives is optional. This will create a point over any position on the surface ($0 \leq x \leq 1$, $0 \leq y \leq 1$).

24.4 SEE ALSO

- *Clip OP* p. 502
- *Divide OP* p. 549
- *Project OP* p. 707
- *Refine OP* p. 719
- *Creep OP* p. 525

24.5 MOUSE AND KEYS -

	Snips and drags away pieces of an object. Click once to select a curve. Click again to cut at the clicked point. Click and drag to cut and move the resulting pieces.
	Same as  , except constrains cutting to the currently located parametric direction.
	Extracts and drags points from a curve or surface. Click once to select a curve. Click again to extract point from clicked location. Click and drag to extract and move extracted piece.

Ctrl **⌘**

Same as **⌘**, except constrains extraction to the currently located parametric direction.

Ctrl **⌘**

Pops-up the Operation menu.

24.6 DESCRIPTION -

This Operation performs a dual function: given a selected curve or surface, it allows you to slice it into two or more pieces with **⌘**, and extract points (if it's a curve or surface) or isoparametric curves (if it's a surface) with **⌘**. The Carve Operation also cuts closed sections. If you click on two edges of a curve you create two closed sub-faces (not two open sub-curves).

First click on the curve or surface you want to operate on to select it. You can skip this step if a primitive of appropriate type is already selected when you enter the Carve Operation. When the cursor is near the selected primitive, the closest point on that primitive is indicated by a small white ring, or dotted isoparametric curve (isoparm). If the ring is not displayed, it means you are too far from the selection. If so, either move the mouse closer to it or click on another curve or surface to select it instead.

Click with the left mouse to cut the selected primitive at the target point. If the primitive is closed, the first cut will open it at that point. Further cutting will split it into two or more parts.

A surface is cut along the isoparametric curve orthogonal to the closest isoparm you have clicked to unless you're holding down **Ctrl**, in which case you are constrained to the currently selected isoparm. The same rule applies to extracting isoparms (using the middle mouse button). Note that cutting happens only along isoparametric curves; houdini does not support arbitrary trim curves yet.

After cutting, the two new primitives (or the opened primitive) remain selected to facilitate further slicing. If you drag the mouse on the Construction Plane before letting go of the left mouse button, you can move the smallest slice away.

Extraction works the same way as cutting does, only it is driven by the middle mouse button (**⌘**). You can extract only points from a curve primitive. To choose between extracting points and isoparms from surfaces toggle the "Extract Points" button. After extraction, the original primitive remains selected to facilitate further extracting. If you drag the mouse on the Construction Plane before letting go of the middle mouse button, you can move the extracted point or isoparm away. Extracted points will not be visible unless you use the Display Dialog to turn them on for display.

The extraction of points and isoparametric curves take into account all the point attributes of the curve or surface extracted from. Thus, attributes such as color, velocity, and event texture coordinates will be extracted along with the positional information.

Dragging the slices or extraction on the floor is subject to the current snapping attractors, if any.

24.7 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog. This displays a blank dialog. The purpose of this is that you are able to call up the parameters dialog, so when you switch to another Operation, it remains and displays the parameters for each Operation as you switch between them.

EXTRACT POINTS

Extracts points instead of carving isoparametric curves from surfaces.

25 CHANNEL OP

25.1 DESCRIPTION

This OP reads sample data from a CHOP and converts it into point positions and point attributes. This makes it complementary to the Geometry CHOP. The Channels created by the Geometry CHOP can be modified and then re-inserted into the OP network via a Channel OP.

This does what a Point OP with a *chopci()* function can do, but is much faster.

By using point groups from the incoming OP, the channels can be inserted only into the group's points.

When you want one component of an array, like the Y value of the point position, you need to put *P(1)* into the *Attribute Scope*.

25.2 PARAMETERS

GROUP

Modify only the points within this point group. If blank, all points are modified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

CHOP NET / CHOP

Specifies which CHOP Network / CHOP contains the sample data to fetch.

METHOD

The sample data fetch method:

Static Uses one channel for each attribute, and all points use this channel (the first point resides at index 0, the next at 1, and so on). The length of the channels should be at least the number of points modified.

Animated Uses one channel per attribute per point. The channels show the animation of each point's position/attribute values.

CHANNEL SCOPE

The names to use for the newly modified attributes. There must be one name per attribute (e.g. an attribute scope of "P" should have a channel scope with three names: tx ty tz).

ATTRIBUTE SCOPE

A string list of attributes to modify in the OP.

list of common attributes:

P	Point position (X, Y, Z) – 3 values
Pw	Point weight – 1 value
Cd	Point color (red, green, blue) – 3 values
Alpha	Point alpha – 1 value
N	Point normal (X, Y, Z) – 3 values
uv	Point texture coordinates (U, V, W) – 3 values

See *Geometry Types > List of Attributes* p. 236 for a complete listing of Attributes.

25.3 LOCAL VARIABLES

none.

25.4 SEE ALSO

- *Channels > Geometry CHOP* p. 352

26 CIRCLE OP -

26.1 DESCRIPTION

This OP creates open or closed arcs, circles and ellipses.

If two NURBS circles that are non-rational (i.e. their X and Y radii are unequal) are skinned, more ioparms may be generated than expected. This is because non-rational NURBS circles parameterise their knots based on chord length, and the Skin OP must consolidate the total number of knots between the two circles before skinning.

To remedy this, you may want to use a Refine OP, and unrefine the resulting skin, or better yet – before unrefining, start with the same circle and use a Primitive or Transform OP to deform the second copy before skinning.

26.2 PARAMETERS

PRIMITIVE TYPE

Select from the following types. For information on the different types, see the *Geometry Types* section. Depending on the primitive type chosen, some OP options may not apply.

- Primitive
- Polygon
- NURBS Curve
- Bézier Curve

ORIENTATION

The plane on which the circle lies.

RADIUS / CENTER */radx /rady /tx /ty /tz*

The Radius of the Circle in the X and Y directions, and the Center of the Circle in X, Y and Z.

ORDER

If a spline curve is selected, it is built at this order.

DIVISIONS */divs*

The number of edges (points +1) used to describe the circle. This option applies to polygons and imperfect splines. The more divisions a circle has, the smoother it looks. Using three divisions makes a triangle, four divisions a diamond, five divisions a pentagon, and so on. Also, for open arc types, the number of points will equal *Divisions + 1*, and for closed arc types, *Divisions + 2*.

Tip: Creating Triangles – Set the *Divisions* to 3 to create Triangles.

ARC TYPE

Determines how the circle should be drawn. Applies to polygons and imperfect splines only.

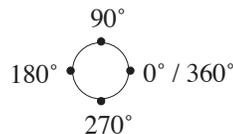
<i>Closed</i>	An enclosed curve.
<i>Open Arc</i>	An open curve segment.
<i>Closed Arc</i>	An open arc with connecting edges to the centre resembling a slice of pie.
<i>Sliced Arc</i>	Same as Closed Arc, but connects every single point to the center of the circle.

Arc options are available for polygonal circles and some spline types. The difference between the choices: *Open*, *Closed*, and *Slice* are illustrated below for the polygonal case:



ARC ANGLES

The beginning and ending angles of the arc. An arc will start at the beginning angle, and proceed towards the ending angle. If beginning=0 and end=360 it will be a full circle. As a reference:



■ *Note: The total angle can exceed 360°, making multiple wraps of the circle.*

IMPERFECT

This option applies only to Bézier and NURBS circles. If selected, the circles are approximated non-rational curves, otherwise they are perfect rational closed curves.

26.3 INPUTS / GEOMETRY TYPES

Primitive, Polygon, Bézier, NURBS curves.



26.4 MOUSE AND KEYS -



Pops-up the Operation menu.

26.5 DESCRIPTION -

This Operation is used to create circles and ellipses. If you click and drag the mouse, it generates a circle whose radii are specified by your drag.

Clicking the mouse button on the Construction Plane without dragging places a circle with radii specified in the Parameters dialog box (default of 1) at the location of the mouse click. The radii of the default circle are aligned with the Construction Plane's X and Y axis.

Typing **Enter** places a circle or ellipse whose size and position are specified in the Parameters dialog. The radii of the default circle are aligned with the Construction Plane's X and Y axis.

If an odd aspect ratio was previously entered in the Parameters dialog, clicking and dragging produces circles which maintain that aspect ratio. This can be reset by clicking on the *Reset Radii* button.

26.6 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog for this Operation. See *Operation Parameters* - p. 496.

PRIMITIVE / POLYGON / NURBS / BÉZIER

Click these buttons to select which type of circle you want to draw. For differences between the four types, see *Geometry Types*.

RESET RADII

Changes the radii of the circle in the parameters dialog back to 1.

26.7 CIRCLE OPERATION MENU -

Display the menu for this Operation by using **Ctrl** + **F11**.

PRIMITIVE / POLYGONAL / NURBS / BÉZIER CIRCLE **E** / **P** / **N** / **B**

These four menu items allow you to specify whether you want to create a circle as a primitive, polygon, NURBS or Bézier curve.

BUILD DEFAULT CIRCLE *Enter*

Draws a circle automatically using the centre and radius values from the parameter dialog, and the circle type from the sub-icons. The radii are aligned with the X and Y axes of the Construction Plane.

26.8 OPERATION PARAMETERS -

RADIUS

These are the X and Y radii of the circle that is placed on the Construction Plane if you click on the Construction Plane without dragging. If you click and drag, the radii of the circle are determined by the drag amount. Entering non-equal values in the XY fields results in elliptical shapes. The X radius is defined by the distance dragged from the centre, while the Y radius varies proportionally with the X/Y radius ratio in the parameter dialog.

CENTER

Determines the location of the center of the circle. This value is updated whenever you click (and drag) to create a circle. A new circle center will be positioned here if you type *Enter*.

ORDER

Sets the spline order when building a circle with a Bézier or NURBS curve type. The lowest order is 2 (linear); the highest is 11. Cubic curves are built by default.

DIVISIONS

The number of *points + 1* used to describe the circle. This option applies to polygons and imperfect NURBS only. The more divisions a circle has, the smoother it looks. Using three divisions makes a triangle, four divisions a diamond, five divisions a pentagon, and so on. Also, for open arc types, the number of points will equal *Divisions + 1*, and for closed arc types, *Divisions + 2*. The number of points on a Bézier circle will be higher than the number of divisions specified, based on the order of the Bézier curve. The # of *Divisions* is ignored when building a perfect (rational) NURBS or Bézier circle.

Tip: Creating Triangles – Set the *Divisions* to 3 to create Triangles.

ARC TYPE

This menu provides you with the choices: *Closed*, *Open Arc*, *Closed Arc*, and *Sliced Arc*. The difference between these is illustrated below:



This option is disabled when building a perfect (rational) NURBS or Bézier circle. To remove a part of the rational curve later, you can use the *Carve OP* p. 486.

The *Closed* and *Closed Arc* options are primarily meant for polygonal circles.

ARC ANGLES

When making an arc rather than a full circle, these values specify the starting and ending points of the arc in degrees. This option is disabled when building a perfect (rational) NURBS or Bézier circle.

IMPERFECT

Specifies whether the NURBS / Bézier circle should be built using rational or non-rational splines. A perfect circle has a rational topology – one that associates non-unit weights with (some) of its vertices. Furthermore, a perfect circle has a predefined number and positions of CVs for any given spline order. An imperfect circle is non-rational and its number of CVs isn't that strictly determined by its order.

Rational circles built this way yield a mathematically perfect shape; however, given their special definition, perfect circles are not always the ideal choice for further modeling of their points. Besides, they represent heavier geometry and may put more pressure both on the CPU and RAM. In practice, you will find imperfect circles to be a better modelling choice, so it is advisable to build perfect circles only when perfect shapes are paramount.

27 CLAY OP

27.1 DESCRIPTION

The Clay OP deforms faces and surfaces by pulling points that lie directly on them. As opposed to the Point OP or other OPs that manipulate control points (CVs), the Clay OP operates on the primitive contours themselves, providing a direct, intuitive, and unconstrained way of reshaping geometry. Thus, rather than translating CVs to change the aspect of the primitive, the Clay OP takes the inverse approach of repositioning the CVs to reflect the tug on the primitive's skin.

The point that defines the area to be modified is called a “target point” or “target” for short. It is expressed as a (u,v) pair in the parametric space of the primitive and ranges between 0 and 1 in both U and V. The image of the target point on the primitive is a 3D point which Clay can displace in several ways. Furthermore, if the primitive is a surface, there are options to pull only the point or a whole isoparametric curve in either U or V.

Clay does not refine the faces and surfaces unless asked to, so the complexity of the geometry does not increase. The area affected by the change varies with each primitive type and topology. In all cases it is possible to reduce to amount of change by inserting a Refine OP before the Clay OP and inserting detail around the target point. For other ways to increase the locality of the deformation as well as its sharpness, see U and V Sharpness below.

If a second input is present, it is possible to snap the target (u,v) point to an (s,t) point on the first primitive of the second input. Without a second input, the primitives can be made to snap to themselves. Moreover, the Clay OP is able to snap the target to arbitrary points in space.

Both this OP and the Align OP can be used effectively as snapping tools and building blocks for curve networks. The main difference between the two OPs is that Clay deforms the inputs partially, while Align translates and/or rotates the whole primitive.

The Clay OP accepts a mix of any combination of face and surface types.

27.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

U

Determines whether the clay operation affects the U parametric direction. Both faces and surfaces respond to this option. If the primitives are face types and the tog-

gle is *off*, clay doesn't change the inputs at all. If the inputs are surfaces and U is *off*, clay will pull the surfaces along the entire V direction (if V is *on*) or not at all.

V

Determines whether the clay operation affects the V parametric direction of the surface. If the inputs are surfaces and V is *off*, clay will pull the surfaces along the entire U direction (if U is *on*) or not at all. If both U and V are *on*, clay will pull the surface at the target (U, V) point specified in the U and V folders below. This option does not affect face types.

27.3 PARAMETERS – U PAGE

U */u*

This value defines a point in the parametric space of a face, representing the location to be affected by the clay operation. This location is referred to as the “target”. For surfaces, U defines a line of constant value in the parametric space of the primitive and requires a second coordinate – V – to specify a unique location (see the V Folder below). Spline faces and surfaces have explicit parametric spaces known as domains; since domains are not restricted to the unit range [0, 1], the Clay OP maps U to the real domain value of the primitive. For polygons and meshes, U is expressed in terms of the number or vertices and columns respectively and in terms of their relative positions.

U BIAS */ubias*

Indicates whether the clay algorithm should use the bias it thinks works best for the given U parameter, or take the value explicitly Operationd beside the toggle. Since clay affects the CVs in the neighbourhood of the given parametric location, the bias can influence the amount of pull applied to the CVs on either side of this location. The effect is a slant of the “wave” the parametric point rides on – towards one side or the other.

U SHARPNESS */usharp*

This parameter affects only NURBS curves and surfaces. The pull generated by clay on these primitives can be smooth or sharp depending on the position of the target relative to the underlying domain (the farther away from a knot, the rounded the bulge) and the knot multiplicity near the target (the higher the multiplicity the sharper the pull). If the pull is too round or affects too big an area in U, the “U sharpness” parameter can reduce it by inserting one or more knots at the target U value. When the U sharpness is zero no knots are inserted. When the U sharpness is 1, all “degree” knots are inserted and the shape becomes very sharp. The U sharpness varies in discrete steps; the number of steps equals the U degree of the spline.

Note: The range of the *U Sharpness* slider varies with the degree of the spline (i.e. the closer it is to 1, the more knots it adds). Since the number of knots added forms a discrete sequence, the slider will automatically jump to the valid positions.

27.4 PARAMETERS – V PAGE

V */v*

This value affects only surface types. V defines a line of constant value in the parametric space of the surface and together with U specifies a unique location in that space. Spline surfaces have explicit parametric spaces known as domains; since domains are not restricted to the unit range [0, 1], the Clay OP maps V to the real domain value of the surface. For meshes, V is expressed in terms of the number or rows and their relative positions.

V BIAS */vbias*

Indicates whether the clay algorithm should use the bias it thinks works best for the given V parameter, or take the value explicitly stated beside the toggle. Since clay affects the CVs in the neighbourhood of the given parametric location, the bias can influence the amount of pull applied to the CVs on either side of this location. The effect is a slant of the “wave” the parametric point rides on -- towards one side or the other.

V SHARPNESS */vsharp*

This parameter affects only NURBS surfaces. The pull generated by clay on these primitives can be smooth or sharp depending on the position of the target relative to the underlying domain (the farther away from a knot, the rounded the bulge) and the knot multiplicity near the target (the higher the multiplicity the sharper the pull). If the pull is too round or affects too big an area in V, the “V sharpness” parameter can reduce it by inserting one or more knots at the target V value. When the V sharpness is zero no knots are inserted. When the V sharpness is 1, all “degree” knots are inserted and the surface becomes very sharp. The V sharpness varies in discrete steps; the number of steps equals the V degree of the spline.

Note: The range of the *V Sharpness* slider varies with the degree of the spline (i.e. the closer it is to 1, the more knots it adds). Since the number of knots added forms a discrete sequence, the slider will automatically jump to the valid positions.

27.5 PARAMETERS – TRANSFORMATION TYPES

The following describe the four types of transformations the Clay OP can apply to the target (U, V) points or isoparametric curves (U or V). All the transformations are exclusive, and all reduce to a signed translation along a direction. The difference between the various transformations is the way the translation and the direction are specified.

MATRIX

The matrix method relocates the target point based on a transformation matrix. For a description of transforms, see the *Transform OP* - p. 805.

VECTOR

The folder method provides a distance and a vector to translate along.

distance */dist*

Specifies the translation distance along the given vector.

normal

The translation will be performed along the primitive normal at the target (u,v) point if the *Normal* button is *On*. Otherwise an explicit direction is required (see “Direction” below). If the input contains face primitives, the primitive normal is independent of the target (u) point. Primitive normals can be displayed from the Display Options dialog.

direction */dirx /diry /dirz*

When the “Normal” option is off, this parameter determines the orientation of the 3D vector along which to translate the target point or isoparametric curve. The (X, Y, Z) values are normalized automatically.

POINT

The point method provides a 3D point in object space that the target point must snap to.

coordinates */coordx /coordy /coordz*

Specifies the absolute 3D location the (U, V) target must translate to.

PRIMITIVE

The primitive method allows the target point to be snapped to a point on another primitive. Any primitive type is allowed, including metaballs, all quadrics, and even particle systems. The point on the destination primitive is expressed parametrically as a (U, V) pair.

If the Clay OP has a second input, then the destination primitive is going to be the first primitive in that input. If there is no second input, the primitives in the first input will snap to themselves (each face or surface to itself).

u and v */uvsnap1 /uvsnap2*

The two values represent the U and V coordinates of the destination primitive in parametric space. The parametric space is known as the “domain” of a primitive. Both U and V are unit values which the program maps to the real primitive domain accordingly. V is ignored if the primitive is a face, circle, or particle system.

28 CLIP OP

28.1 DESCRIPTION

Cutting and creasing source geometry with a plane.

28.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

KEEP

Options controlling what part of the clip to keep:

Primitives Above the Plane Retain the primitives above the cutting plane.

Primitives Below the Plane Retain the primitives below the cutting plane.

All Primitives Retain both the top and bottom primitives generated by the cutting plane. Creates a crease in the geometry.

DISTANCE */dist*

Allows you to move the cutting plane along the Direction vector. If the direction (plane's normal) is 0, 1, 0, putting a positive number in the Distance field moves the plane up in Y.

DIRECTION */dirx, /diry, /dirz*

The default values of 0, 1, 0 creates a Normal vector straight up in Y, which is perpendicular to the XZ plane, which becomes the clipping plane. 1,0,0 points the normal in positive X, giving a clipping plane in YZ. The plane may be positioned at an angle by using values typed in (1,1,0 gives a 45° angle plane) or interactively by using the direction vector jack.

CREATE GROUPS

When checked, allows you to generate specific groups for the geometry above and below the cutting plane. See the two group option fields below. This option is only available when *All Primitives* are kept.

ABOVE PLANE

When Create Groups is checked, you can assign the geometry below the cutting plane to the Group name typed in this field.

BELOW PLANE

When Create Groups is checked, you can assign the geometry above the cutting plane to the Group name typed in this field.

28.3 INPUTS / GEOMETRY TYPES

This OP accepts one input. The Clip OP will attempt to clip all primitives. If the primitive type when clipped does not destroy the primitives structure, it will be clipped, i.e. polygons, particle systems. Others are deleted if they are intersected by the clipping plane.

28.4 USES / WORKS IN RELATION WITH

This OP can be used to clip out polygons outside the field of view to reduce the number of primitives in a given scene.

29 COMB OP

29.1 DESCRIPTION

The Comb brush is designed to allow you to intuitively adjust surface point normals in the Viewport. As its name suggests, the brush is often used to initialize the normals for hair growth operations.

By dragging the brush across a surface with point normals, the normals will tilt in the direction of the drag.

29.2 PARAMETERS – OPERATION PAGE

OPERATION

Allows you to: *Comb Normals*, *Smooth Normals*, or *Erase Changes*.

COMB LIFT *//lift*

This parameter controls how far the normals will be lifted off the surface. A value of 1 will cause them to move to an upright position.

KEEP NORMAL LENGTH

If not set, the normals will be normalized to unit length as they are combed.

If set, the length of the normals will be left unchanged by the comb operation.

This flag also affects smoothing of normals – if set the normals will not be normalized after smoothing.

OVERRIDE NORMAL

This allows you to specify an attribute other than N to comb. The attribute must be 3 floats in length.

RESET ALL CHANGES

This button will restore the geometry to its original state.

29.3 PARAMETERS – BRUSH PAGE

SHAPE

The basic shape of the brush: circle, square, or bitmap.

BITMAP

What bitmap to use. The alpha channel becomes the brush.

RADIUS */radius*

The radius of the brush.

BRUSH ANGLE */brushangle*

How far to rotate the brush.

BRUSH SQUASH */squash*

Amount to squash the brush in the y direction before rotation.

OPACITY */opacity*

The amount to affect the stencil mask.

BRUSH SPLATTER */brushsplatter*

A random noise in the brush's opacity based upon the position on the brush.

PAPER GRAIN */papergrain*

A random noise on the object's stencil mask based on the object position.

SOFT EDGE */softedge*

Percentage of the brush to be rolled off.

KERNEL FUNCTION

Which metaball kernel to use for the roll off.

UP VECTOR TYPE

How the brush should be oriented on the surface:

Stroke Direction Oriented in the direction in which the brush moves.

Fixed Oriented as specified in the Up Vector field.

UP VECTOR */upvector*

The fixed up vector to orient brush to.

29.4 PARAMETERS – STROKE PAGE

ORIENT BRUSH TO SURFACE

Switches between the brush being perpendicular to the surface or always oriented along the view direction. If you are having trouble with a shaky brush, try turning this off.

REALTIME MODE

Set this to paint geometry as it's deforming.

The last few parameters on this page are read-only parameters. While it is theoretically possible to manually apply brush strokes by adjusting these parameters, that would be excruciatingly difficult. Their primary use is to provide feedback as to where the brush is currently hitting your geometry. This can be used as a sort of 'geometry eyedropper' when you want to quickly investigate some troublesome geometry.

DIRECTION

This is the normal on the hit primitive at the hit position.

HIT LOCATION

This is the position in space where the brush ray hit the geometry.

HIT PRIMITIVE

This is the primitive number that was hit by the brush ray.

HIT UV

These are the parametric coordinates where the primitive was hit. They are the UV coordinates used to evaluate the surface with a prim call, for example, and is not to be confused with any "uv" texture coordinates that may be present.

EVENT

This is used internally to track the current state of the drawing mode. No-op is the default and is used to prevent any accidental writes using out of date or irrelevant hit information.

29.5 SEE ALSO

- *Capture Paint OP* p. 480
- *Paint*
- *Sculpt OP* p. 735

30 CONSTRUCTION PLANE OPERATION -

30.1 MOUSE AND KEYS -

	Drag to move the Construction Plane along its own XY plane.
 	Drag to move the Construction Plane up with respect to itself, in the direction of the up vector.
 	Drag to move the Construction Plane along its own XY plane constrained to the nearest 45°.
	Drag to build a plane perpendicular to the current plane. Use  to constrain to 45°, and  to move in the direction of the up vector.
	Click once, twice, or three times as described in the section below to define a new plane.
	Click once, twice, or three times as described in the section below to define a new up vector.
 	Pops-up the Operation menu.

30.2 DESCRIPTION -

The Construction Plane is a grid-like tool to assist you in creating and translating geometry. All geometry in Model Mode is built and edited relative to the Construction Plane, which is used as a point of reference against which objects are moved: either along its plane, or upwards from it along the up vector (see *Up Vector* p. 405). In the process of manipulating geometry, it is often necessary to move, rotate, and resize the Construction Plane relative to other geometry elements. These functions are achieved using this Operation. For more info, see *Construction Plane* p. 404.

UP VECTOR

The *Up Vector* determines the direction in which objects move when you  drag them. Its orientation is indicated by the little pink vector located at the origin of the Construction Plane (See *Up Vector* p. 405 for illustration). Initially, this vector is obstructed by the red segment representing the plane normal.

INTERACTORS

Interactors are items that can be clicked in the Model Editor to define a new place or orientation for the Construction Plane. As a rule of thumb, Interactors are points on any object in the Model Editor window, except primitive geometry elements. More specifically, Interactors are any of the following:

- Points, which may include vertices of the geometry primitives, end-points of handles, and intersections of guide edges.
- Construction Plane points and grids.
- Edges, which may include edges or line segments of geometry primitives, handles, guide edges, and the Construction Plane grid.
- Normals, which include: point and primitive normals.

30.3 PLANE OPERATIONS -

The Construction Plane can be repositioned and reoriented using the mouse, the keyboard, or the parameter dialog. Two ways of editing the Construction Plane are: by *dragging* with either \square or \square to shift the plane or build one perpendicular to it, and by *clicking* with \square or \square to specify points or vectors that define a new plane.

DRAGGING

\square	Move the plane along itself.
$\text{Alt} \square$	Move the plane along the up vector.
$\text{Ctrl} \square$	Move the plane along itself constrained to 45°.
$\text{Ctrl} \text{Alt} \square$	Move the plane along the up vector constrained to 45°.
\square	Drag to build plane perpendicular to the current plane. X and Y size of new plane are determined by the drag. Ctrl and Alt constrain and move along the up vector.

CLICKING

(p_n = click a point interactor, e_n = click an edge interactor)

$\square p_1$, then Enter	Define new plane origin without changing orientation.
$\square p_1$, $\square p_2$, then Enter	$p_1 - p_2$ = plane X-axis, orientn. changes, but not origin.
$\square p_1$, $\square p_2$, $\square p_3$	Define new plane. p_2 = new plane origin, $p_2 - p_3$ = new plane X-axis. $p_1 - p_3$ cannot be co-linear because a line cannot uniquely define a plane.
$\square e_1$, $\square p_1$	e_1 = plane X-axis, p_1 completes 3-point plane defn.
$\square e_1$, $\square e_2$	Common point = plane origin. e_2 = plane X axis. e_1 and e_2 must be adjacent.
$\text{Alt} \square p_1$, $\text{Alt} \square p_2$	$p_1 - p_2$ = new plane normal.
$\text{Alt} \square e_1$	e_1 = new plane normal.
$\text{Alt} \square$	any point or primitive normal = new plane orientation. origin remains unchanged.

30.4 UP VECTOR OPERATIONS -

Use the middle mouse (MB) to perform the following operations.

MB p_1 , MB p_2 , MB p_3 Up vector is perpendicular to imaginary surface created by p_1 - p_2 - p_3 .

Alt MB p_1 , Alt MB p_2 $p_1 - p_2 =$ up vector direction.

Alt MB p_1 , MB p_2 $p_1 =$ up vector origin. $p_1 - p_2 =$ direction of up vector.

Alt MB e_1 up vector = e_1 .

MB any point or primitive normal = up vector.

30.5 SUB-ICONS -

30.6 CONSTRUCTION PLANE MENU -

Call up the menu for this Operation by using Ctrl MB .

ERASE C-PLANE ELEVATION E

Drops the elevation previously created by an Alt drag or snap to zero. You will now be working directly on the Construction Plane.

FLIP C-PLANE ELEVATION M

Mirrors the elevation symmetric to the plane.

FLIP C-PLANE NORMAL N

Changes the Construction Plane's normal so it points in the opposite direction.

INCREASE GRID IN X AND Y +

Grow the Construction Plane by one grid unit in all directions.

DECREASE GRID IN X AND Y -

Shrink the Construction Plane by one grid unit in all directions.

INCREASE GRID ALONG ITS POSITIVE X / Y > / $\text{Shift}>$

Grows the number of columns by one in the positive X / Y axis of the plane.

INCREASE GRID ALONG ITS NEGATIVE X < / $\text{Shift}<$

Grows the number of columns by one in the negative X / Y axis of the plane.

ALIGN C-PLANE WITH VIEWING PLANE

Repositions the Construction Plane so that it is parallel to the plane created by the Viewport window.

ALIGN NORMAL WITH WORLD X / Y / Z AXIS / /

Aligns the Construction Plane's normal with the world X/Y/Z axis.

CENTER C-PLANE AROUND WORLD X / Y / Z AXIS / /

Resets the Construction Plane to one of its default locations and orientations, centred at the origin of the world space and aligned with the world axes.

ALIGN UP VECTOR WITH C-PLANE NORMAL

Positions the up vector such that it matches the Construction Plane's normal.

ROTATE C-PLANE TO FRONT/ BACK/ LEFT/ RIGHT PLANE / / /

Rotating to the front or back flips the plane 90° forward or backward about the center of the X axis. (Same function as *X-rotate* sub-icon.)

Rotating to the left or right flips the plane 90° right or left about the centre of the Y axis.

30.7 OPERATION PARAMETERS -

GRID SPACING

Determines the spacing of each slot of the Construction Plane grid in X and Y.

GRID COUNT

Determines the number of rows and columns in the Construction Plane grid.

GRID RULER

Entering values in these fields uniformly divides the Construction Plane grid according to your specifications. For example, entering a value of five in each of these fields produces a division of the Construction Plane grid that measures five units by five units.

GROW FROM ORIGIN

Grows the grid from the origin at the corner of the Construction Plane, rather than from its center.

ORIGIN

The XYZ coordinates of the origin of the Construction Plane in world coordinates.

NORMAL

The normal is a vector located at the Construction Plane's origin, which is perpendicular to the Construction Plane. Entering values here defines a new value in world co-ordinates for this normal. Because the normal must always remain perpendicular to the Construction Plane, entering a new value here also re-orientes the Construction Plane. The operation is somewhat analogous to steering a ship by its rudder.

UP-VECTOR

The up-vector determines the direction things move when you **Alt** drag them. This is usually perpendicular to the Construction Plane. In some cases it is desirable to have the up-vector go in some other direction. Selecting from the menu below (*Lock Up-Vector To:*), and entering in values here, define a new direction for the up-vector independent of the Construction Plane.

LOCK UP-VECTOR TO

Usually, it is desirable to have the up-vector perpendicular to the Construction Plane (identical to the plane normal). This menu allows you to lock it to the plane normal, to one of the world axes, or to a specific direction.

31 CONVERT OP

31.1 DESCRIPTION

This very powerful OP converts geometry from one geometry type to another type.

31.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

FROM TYPE

Determines which geometry by type will be converted. The default is All Geometry:

- All Types – all geometry will be converted
- Circle
- Sphere
- Tube
- Particles
- Meta-ball
- Polygon
- Mesh
- Bézier Curve
- Bézier Surface
- NURB Curve
- NURB surface
- Pasted Surface

CONVERT TO

Determines what the above From Type geometry will be converted to. Conversion to polygons is the default:

- Polygon
- Mesh
- Bézier Curve
- Bézier Surface
- NURB Curve
- NURB Surface
- Pasted Surface
- Circle
- Trimmed NURBS Surface
- Trimmed Bezier Surface

When set to *Divisions per Span*, specifies the number of divisions per span.

Tip: Animate the *Level of Detail* based on how close your character or geometry is to the camera by using the *primdist()* expression. Then the detail will increase/decrease as the camera gets closer/further away.

trim curve

The trimmed part of a surface will be converted using this Trim LOD (level of detail) instead of using an implicit “1” constant.

U / V ORDER */orderu /orderv*

When converting to a spline type, this specifies the *degree + 1* of the U or V basis function.

PASTE COORDINATES

from feature surfaces

The resulting mesh will have the shape of the paste hierarchy (i.e. the top-most features will determine the shape).

from base surface

The resulting mesh will take the shape of the bottom-most surface.

PASTE ATTRIBUTES

from feature surfaces

Each mesh point has the attributes of the top-most feature at that location.

from base surface

The resulting mesh will inherit the primitive attributes of the root surface (e.g. the material), and the point attributes will be those of the root surface as well.

By using base coordinates and feature attributes you are, in fact, pasting attributes onto the surface.

PRESERVE ORIGINAL

When checked, the original geometry will be retained along with the converted geometry.

INTERPOLATE THROUGH HULLS

This option applies to the conversion between polygonal faces and grids to NURBS and Bézier surfaces and curves. When selected, the resulting curves retain the same topology as the original polygon. When not selected, the polygon points are used as a hull to define the new curve or surface.

31.3 INPUTS / GEOMETRY TYPES

This OP accepts only one input, of any geometry type.

Note: When a primitive is converted, it retains the same primitive and group attributes as the original.

31.4 USES / WORKS IN RELATION WITH

Certain tools work better on one geometry type than another. This OP is good for converting one type of geometry to another for speed in animation.

31.5 NOTES

FACE TO SURFACE CONVERSION

When converting from a set of polygons to a mesh, a single mesh will result only if:

- more than one polygon is in the input, and
- each polygon has exactly four points, and
- the polygons are arranged as n rows by n columns
- the polygons share coincident points (see *Facet OP* p. 570)

otherwise, each polygon is converted individually into a mesh. In fact, any individual face can be converted to any surface. This is accomplished by cutting the face into three or four adjacent sections, and then creating a patch from them.

32 COOKIE OP

32.1 DESCRIPTION

The Cookie operation applies a cookie cutter between two polygonal objects which may be open or closed. Cookie determines inside and outside by looking at the polygons normals. The normals are assumed to point to the outside of the object.

The polygons need to be planar for this to produce accurate results. If point or vertex attributes are present, they need to have at most four sides to be able to interpolate those attributes.

32.2 PARAMETERS

GROUP A / B

The polygons to form the A / B object.

3D TOLERANCE */tol3d*

The distance to determine if polygons are degenerate.

PRE-CONVEX GEOMETRY

Convexes the incoming geometry to ensure all polygons are planar. If point or vertex attributes are present, it also ensures maximum number of sides on a polygon is four.

32.3 PARAMETERS – BOOLEAN TAB

This does boolean style operations.

OPERATION

<i>Union</i>	Remove the interior of the two.
<i>Intersect</i>	Remove the exterior of the two.
<i>A minus B</i>	Cut B out of A.
<i>B minus A</i>	Cut A out of B.
<i>User</i>	Use the toggles to determine what is kept.

KEEP INSIDE A

Keep all of the A object inside of the B object.

KEEP INSIDE B

Keep all of the B object inside of the A object.

KEEP OUTSIDE A

Keep all of the A object outside of the B object.

KEEP OUTSIDE B

Keep all of the B object outside of the A object.

CHECK FULL ENCLOSURE

Cookie usually determines whether a polygon is inside or outside the other object's geometry by looking how that polygon, or it's neighbours, intersected with the geometry. However, if a polygon is not connected to any other polygon, an explicit ray cast needs to be made to determine the side. This flag enables that ray cast.

32.4 PARAMETERS – CREASE TAB

This generates the polygonal crease between the two pieces of geometry.

KEEP SOURCE

Determines if the original geometry should be included in the output.

AUTOJOIN CREASES

Tries to connect the crease segments into long continuous polygons.

32.5 PARAMETERS – JITTER OPTIONS

DO JITTER

If checked, the second input is "jittered" by a small amount before intersection, and unjittered afterwards. Turn this on if you have coincident faces which prevent you from cutting properly.

JITTER SEED */seed*

This is the random number seed for the jitteramount.

JITTER AMOUNT */jitteramount*

This is the scale of the jitter variable.

32.6 SEE ALSO

- *Clip OP* p. 502
- *Surfsect OP* p. 788

33 COPY OP

33.1 DESCRIPTION

This OP lets you make copies of the geometry of other OPs and apply transformations to the copies.

It also allows you to copy geometry to points on an input template.

33.2 PARAMETERS – COPY PAGE

SOURCE GROUP

Specifies a subset of input primitives to copy from. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

TEMPLATE GROUP

Specifies a subset of template primitives from which to copy onto. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

OF COPIES

- i) Sets number of Copies to be made of the source.
- ii) For a template input, it specifies the number of copies to be placed at each point of the template.

ROTATE TO NORMAL

Only used when a template input is specified. If the template is a sphere, and the first input is a circle, a circle is placed at each point of the sphere. With this option on, all the circles will re-orient to face the surface of the sphere (a default sphere has normals radiating outwards from the centre).

If an up attribute exists on the template geometry, then this will be used (along with the normal) to fully define the rotates for the copies. An up attribute is created with the Point OP.

TRANSFORM CUMULATIVE

Each transformation “builds” on the location left by the one before it. Transformations are cumulative as the Copy OP produces new copies.

TRANSFORM ORDER

Sets the overall transform order for the transformations. The transform order determines the order in which transformations take place. Depending on the order, you can achieve different results using the exact same values. Choose the appropriate order from the menu.

ROTATE ORDER

Sets the order of the rotations within the overall transform order.

TRANSLATE, ROTATE, SCALE, PIVOT

These allow you to specify the Translation (how much it moves over in a given direction), Rotation, and the Scale between each copy. Three columns are given for X, Y, and Z coordinates. Guide geometry is provided for the Pivot's translations. The Pivot is represented by a single red dot in the Viewport. Changing the Pivot parameters moves this point of reference.

UNIFORM SCALE

Uniform Scale allows you to shrink or enlarge geometry along all three axes simultaneously.

NORMALS MAINTAIN LENGTH

Vector type attributes (i.e. normals, velocity) maintain the same length under transforms. i.e. When geometry is scaled, the normals remain constant in length.

CREATE OUTPUT GROUPS / COPY GROUPS

If selected, this creates a group for each copy number, and places each primitive created at that stage into it.

An entry in the *Copy Groups* field defines the base name of the groups created.

33.3 PARAMETERS – STAMP PAGE

Stamping is allowed in any parameter in Houdini. The only requirement is that the stamped parameter is upstream in some fashion from the Copy OP doing the stamping.

STAMP INPUTS

When enabled, it will Stamp preceding variables for each input copied.

PARAM I - 10

Token and value of each stamp variable.

These change the values of parameters which include *param()* function.

See the example, below.

33.4 PARAMETERS – ATTRIBUTE PAGE

This page allows you to determine how point attributes on template geometry affect attributes on the source geometry. The template attribute can modify the source in four ways:

<i>Set</i>	Override the source attribute.
<i>Multiply</i>	Multiply the source attribute.
<i>Add</i>	Get Added to the source attribute.
<i>Sub</i>	Get Subtracted from the source attribute.

The template point attributes are able to affect point, primitive, or vertex attributes in the source geometry simply by entering values in the appropriate fields.

EXAMPLES**scenario 1**

The geometry to be copied has prim. colors, the template geometry has point colors.

By setting the *Multiply* column of the *To Primitive* field to the value of “Cd”, the point colors in the template will multiply the prim. colors of the source geometry. Each copy will have its colors multiplied by the template point that it’s copied to.

scenario 2

A simple grid has texture coordinates, but it’s desired to have each copy of the grid offset differently into the texture map.

By assigning a “uv” attribute to the template geometry, then setting the *Add* column of the *To Point* field to the value of “uv”, the template texture coordinates will be added to the original grid’s texture coordinates.

scenario 3

It is often desirable to take the point color and velocity attributes of a particle system and assign them to the point/primitive attributes of geometry which we copy to each particle. This is trivial since we simply set the *Set* columns of the *To Prim* or *To Point* fields to the desired attribute patterns.

33.5 INPUTS / GEOMETRY TYPES

COPY DATA

The OP you wish to make Copies of.

TEMPLATE

This input is optional. If specified, a copy of the Copy Data is placed at each point in the template.

33.6 LOCAL VARIABLES

CY	The Copy Number.
NCY	The Number of Copies.
PT	The Point Number of the Template.
NPT	The number of points in the Template.
AGE	The number of seconds a particle in the template has been alive.
LIFE	The ratio of the age of a particle to its expected life. This is a number between 0 (birth) and 1 (death).
ID	The ID of the particle being sent to the Copy OP through its template input.

VARIABLES WHICH REFER TO TEMPLATE GEOMETRY

The following variables refer to the template geometry if there is a template as an input. Otherwise they refer to the input geometry:

CEX, CEY, CEZ	Defines the centroid of the geometry.
XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX	The extents of the bounding box around the geometry.
BBX, BBY, BBZ	The relative position of the point with respect to the bounding box of the geometry.
SIZEX, SIZEY, SIZEZ	The size of the bounding box of the geometry.

33.7 TIPS

- You can use the *NCY* local variable to calculate the degrees of rotation for a given number of copies. For example, if you have 28 copies, you can set the rotation to be: $360/\$NCY$ – this would automatically give you 12.8571 degrees, evenly spacing your 28 copies around the full circumference of the circle.
- Using a Particle OP as the Template object, you can copy objects defined in the Copy Data input to each particle template. This allows you, for example, to copy a Bee to each particle to create a swarm of bees – although you may opt to do this using Particle *Geo Instancing* in order to save on memory.
- Make a series of copies about an axis, and skin them to achieve lathe-like effects, similar to the results achieved with the Revolve OP.

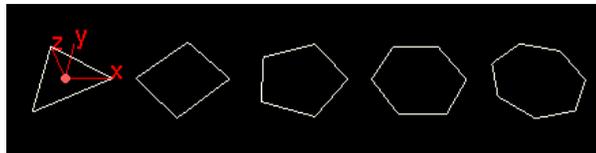
33.8 EXAMPLES

CREATING STAMPED GEOMETRY

1. Place a Circle OP, and set its type to *Polygon*.
2. Set the number of *Divisions* in the Circle to: $param("sides",3)$
The function $param()$ returns the value of the global parameter *sides*.
If it is not yet defined it will return a value of 3. (*Be sure to enable Expressions!*)
3. Append a Copy OP, and set the *Number of Copies* to 5;
and set *Translate X* to: 2.5 .
4. In the *Stamp* page of the Copy OP, turn on *Stamp Inputs*.
Set *Param 1* to: $sides - \$CY+3$.



5. This creates a triangle on the first stamped copy; a square on the next; a pentagon on the third, and so on. The geometry for each copy is cooked separately.



You can set multiple stamp *Params* at once and they can be used anywhere in the ancestry of the copy's input.

OTHER EXAMPLES OF STAMP PARAMETER

Take a look in: *Ref > CHOPs > Sea Snakes CHOPs Demo p. 257*.

34 CREASE OP

34.1 DESCRIPTION

Crease adds the *creaseweight* attribute to polygon edges for use with the Subdivide operation. The group field expects a string as follows, each word separated by a space:

<i>n</i>	<i>n</i> is an Integer	Creases all edges in primitive <i>n</i> .
<i>pa</i>	<i>a</i> is an Integer	Creases all edges with point <i>a</i> .
<i>nem</i>	<i>n</i> and <i>m</i> are Integers	Creases edge <i>m</i> of primitive <i>n</i> .
<i>pa-b</i>	<i>a</i> and <i>b</i> are Integers	Creases edge between points <i>a</i> and <i>b</i> .

34.2 PARAMETERS

GROUP

The edges to crease.

OPERATION

Add to Existing Value

Accumulates the crease value.

Set

Replaces existing crease with specified one.

Delete

Removes crease value.

CREASE

/crease

The Crease value.

ADD VERTEX COLOR

Adds colour to creased vertices to give visual feedback.

34.3 SEE ALSO

- *SubDivide OP* - p. 777

35 CREEP OP

35.1 DESCRIPTION

This OP lets you deform and animate Source Input geometry along the surface of the Path Input geometry.

35.2 PARAMETERS

INITIALIZE

The Source Input is Translated, Rotated and Scaled so as to complete the given options listed below.

<i>Fill Path</i>	Values are computed that stretch or shrink the input geometry to the full length and width of the Path geometry. These values are placed in the nine transform fields at the bottom of the OP.
<i>Keep Proportions</i>	Similar to above, but the values are initialized so as to minimize distortion of object geometry.

TRANSLATE

/tx /ty /tz

Translate the Source Input Creep geometry on the surface of the Path Input.

<i>/tx</i>	Translate the Source Input along the Path Input's U parameter
<i>/ty</i>	Translate the Source Input along the Path Input's V parameter
<i>/tz</i>	Translate the Source Input along the Path Input's W parameter (elevation above/below Path)

ROTATE

/rx /ry /rz

Rotate the Source Input creep geometry on the surface of the Path Input.

<i>/rx</i>	Rotate the Source Input along the Path Input's U parameter
<i>/ry</i>	Rotate the Source Input along the Path Input's V parameter
<i>/rz</i>	Rotate the Source Input along the Path Input's W parameter

SCALE $/sx /sy /sz$

Scale the Source Input creep geometry on the surface of the Path Input.

$/sx /sy /sz$ Scale the Source Input along the Path Input's U / V / W parameters respectively.

The above three transform fields set the translation of the input geometry along the rows (U) and columns (V) of the Path. A creep translate of 0.5, -0.5, 1 puts the source in the middle of the path, one unit away from it.

For scaling, a source of 0.2×0.3 in size will span 20% of the columns, and 30% of the rows on the Path.

35.3 INPUTS / GEOMETRY TYPES

The geometry of the Source OP is crept along the path of the Path OP. The path type can be any primitive, but should define a surface (e.g. NURBS surface, primitive tube, etc.). Curve paths (circles, polygons, etc.) will squash the source geometry's height. This OP will transform all vector attributes, even of particles – the velocities will be altered to reflect the transformation that creep does.

Note: NURBS surfaces might distort the crept geometry due to the non-uniformity of their U and V knot sequence. If the knots are not laid out uniformly (as in the case of a NURBS spline), knots closer together will squash the crept geometry in the corresponding surface region.

The Grid, Skin, Sweep, Tube, and Sphere (using Unique Points at Poles if polygons are used) OPs are some of the OPs that produce usable Meshes.

The Source Input OP's geometry is wrapped onto the Path's, and the position and orientation of the Source on the Path is controlled by the nine fields that appear at the bottom of the OP: Translate, Rotate, and Scale XYZ.

The Z translate, Z scaling and X/Y rotation of the Source to the Path depends on the surface normals of the Path's geometry.

35.4 USES / WORKS IN RELATION WITH

Creeping the starting profile for a filleted tube onto a surface can be accomplished with this OP. Use a Transform OP or Copy OP to scale and translate the crept profile generating a series of cross-sections to skin with the Skin OP.

36 CURVE OPERATION -

36.1 DESCRIPTION

The Curve OP exists for creating polygonal, NURBS or Bezier curves from a given list of coordinates. These coordinates can be absolute, relative or they can reference a point in existing geometry (from the option input). The points can be interpreted as CVs, Breakpoints or Freehand and options exist for fitting settings.

The optional input allows the addition of geometry to which the curve can snap (see the *Coordinates* parameter (below) for more information).

36.2 PARAMETERS

PRIMITIVE TYPE

Type of geometry created.

METHOD

How the list of coordinates are to be interpreted.

COORDINATES */coords*

List of coordinates with which to build the curve.
Coordinates can be specified in the following formats:

x,y,z	x, y and z are floating point numbers. The point at these coordinates will be assigned a default weight of 1.
x,y,z,w	x, y, z and w are floating point numbers.
@ x,y,z	The point at these coordinates will be located relative to the coordinates specified before it, or 0,0,0,1 if none exist, and will be assigned a default weight of 1.
@ x,y,z,w	The point at these coordinates will be located relative to the coordinates specified before it, or 0,0,0,1 if none exist. The weight specified will be relative to the previous point's weight.
p_n	n is an integer (e.g. $p_0, p_1...$). This will reference a point n from the input geometry.
P_n	n is an integer (e.g. $P_0, P_1...$). This will copy a point n from the input geometry.

`xbu | xb[u]`

`u` and `x` are integers (e.g. `0b0`, `0b[1]...`). Creates a point at breakpoint `u` of primitive at index `x`. The primitive must be a curve.

`xb[u,v]`

`u,v` and `x` are integers (e.g. `0b[0,0]...`). Creates a point at breakpoint `(u,v)` of primitive at index `x`. The primitive must be a surface.

Note: Each set of coordinates must be separated by a space.

36.3 PARAMETERS – CURVE PROPERTIES PAGE

CLOSE */close*

Toggles between creating an open or closed curve.

NORMALIZE BASIS

Enables or disables normalizing of the curve's basis.

ORDER */order*

The order of NURBS or Bezier curve.

PARAMETERIZATION */param*

Parameterization method.

36.4 PARAMETERS – FITTING PROPERTIES PAGE

TOLERANCE */tolerance*

Precision of fitting algorithm.

SMOOTHNESS */smooth*

Curvature of the curve.

PRESERVE SHARP CORNERS */keepshape*

Enables or disables fitting of sharp corners.

KEEP INPUT GEOMETRY

Toggles whether input geometry is copied.

36.5 SEE ALSO

- *Add OP* p. 426
- *Fit OP* p. 577

36.6 MOUSE AND KEYS -

<p> or </p> <p></p> <p>  or  </p> <p> </p> <p> or </p> <p> </p>	<p>Place points, drag to pull out curves, or drag only for freehand curves (select with the menu).</p> <p>Finish and select curve.</p> <p>Elevate cursor in direction of Up Vector.</p> <p>Constrain cursor to 45° from current cursor position on current Construction Plane. Hold down  continuously to constrain each node at 45° angles relative to the previous click.</p> <p>Backs up one step, and removes the last entered node.</p> <p>Pops-up the Operation menu.</p>
--	--

36.7 DESCRIPTION -

Use this Operation to draw polygons, NURBS, and Bézier curves. Type the , , or  keys respectively, or click on the sub-icons to specify the curve type.

Once you've selected the type of curve you want to draw, all points are placed in the same way, based on the chosen construction style (CVs, Breakpoints, or Freehand). Use the  key while drawing to toggle the curve open or closed, or enable the check-box button *Open* as desired.

Clicking with the  mouse or typing  completes the curve.

ELEVATION OF PLACED POINTS

The points are drawn on the Construction Plane, or a distance above or below it. If you have selected a point prior to starting a new curve, new points adopt the same elevation from the Construction Plane as the previously selected point.

To change the elevation (height above the Construction Plane) drag the mouse while you holding down the  key to define the new altitude.

HOW TO ADD POINTS TO AN EXISTING CURVE

To add points to the end of an existing curve, follow these steps:

- select the curve you wish to extend (only one) in the Select Operation
- enter the Curve Operation
- -click near the end you wish to extend



- add new points as desired
- click with  or type *Enter* to finish

Not only can you add points to the existing curve; you can also open it, close it, or even delete old points.

You can extend a curve only with CV style editing.

36.8 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog for this Operation. See *Parameters* - p. 532.

POLYGON / NURBS / BÉZIER

Click these buttons to select which type of curve you want to draw. For differences between the three types, see *Geometry Types*.

CONSTRUCTION STYLE MENU

Choose one of: CVs, Breakpoints, or Freehand.

<i>CVs</i>	Click where you want the control points to be placed. You need at least <i>order</i> CVs for an open spline (NURBS, Bézier), and <i>order-1</i> if closed.
<i>Breakpoints</i>	Click on points that you want the curve to go through. Two breakpoints are sufficient to build a valid curve. This style is synonymous with the CV style if you are building polygons. See the <i>Geometry Types</i> section for more information about breakpoints.
<i>Freehand</i>	Click on the Construction Plane and drag while drawing the desired shape. The curve is completed when you release the mouse button. If creating a NURBS or Bézier curve, it will become smoothed when the mouse button is released.

OPEN / CLOSED CURVE

Determines whether the curve will be open or closed. Open curves will not have their end point connected to the starting point; closed curves will automatically have their last point connected to their first point.



36.9 CURVE OPERATION MENU -

Call up the menu for this Operation by using **Ctrl** .

POLYGONAL / NURBS / BÉZIER CURVE **P** / **N** / **B**

These three menu items allow you to specify whether you want to create a curve (curve) as a polygon, NURBS or Bézier.

TOGGLE OPEN / CLOSE **O**

Toggles curves to be open or closed. You can toggle this while building the curve.

CREATE WITH CVS **V**

When selected, you can create a curve by clicking where you want the control points to be placed. You need at least *order* CVs for an open spline (NURBS, Bézier), and *order-1* if closed.

CREATE WITH BREAKPOINTS **K**

When selected, you can create a curve by clicking on the points you want the curve to go through. Two breakpoints are sufficient to build a valid curve. This style is synonymous with the CV style if you are building polygons. See the *Geometry Types* section for more information about breakpoints.

FREEHAND DRAW **F**

When selected, you can create a curve by clicking on the Construction Plane and holding down the mouse button while drawing the desired shape. The curve is completed when you release the mouse button. NURBS and Bézier curves will become smoothed when the mouse button is released.

TRANSLATE C-PLANE TO LAST POINT IN CURVE **T**

Moves the Construction Plane so its origin is on the last point in the curve that you are drawing.

FINISH BUILDING **Enter**

Completes drawing the curve, when drawing with CVs or Breakpoints.



36.10 PARAMETERS -

ORDER

Allows you to set the order of the curve (NURBS or Bézier). The order ranges from 2 (linear) to 11. Cubic curves are built by default. See the *Geometry Types* section for a discussion of order.

NEXT POINT

Allows entry of XYZ coordinates for the next point of the curve. This is more accurate than entering the points in with the mouse only. Once you have entered the values, click the *Add Point* button to accept the values.

WEIGHT

Specify the weight of a point in a curve by entering a value here. Instead of typing, you can use the slider to enter values quickly. The weight of a point determines how much “pull” that node will have on a Bézier or NURBS curve (see *Geometry Types* for details).

ADD POINT

Accepts the values entered in the *Next Point* edit fields, and places the next point of the curve at that location.

DELETE LAST POINT

Takes back the last point you entered. Equivalent to the **⌫** key.

PARAMETERIZATION

Specifies the parameterization of the data. Only applicable for the *Breakpoints* style.

Uniform

Uniform parameterization uses equally spaced parameter values. It works best when the geometry is very regular. When the data is unevenly spaced, this approach can produce very unintuitive shapes, and is not recommended.

Chord Length

Chord length computes the parameterization of the data based on the relative distances between successive data points. It is the most commonly used approach because it tends to produce the most accurate results.

Centripetal

Centripetal parameterization is similar to chord length, but yields better results when the data has very sharp corners.

TOLERANCE

Only available with the *Freehand* style. This is the primary precision factor when approximation fitting the freehand curve. The smaller the tolerance, the closer the fit and the higher the number of generated vertices.

SMOOTHNESS

Only available with the *Freehand* style. For a set tolerance, the smoothness factor allows for more or less roundness in the generated shape. If this parameter is zero, it does not mean that the fit will be sharp. It simply indicates that no additional smoothing is required past the level of smoothness already achieved with the given tolerance.

SHARP CONNECTIONS

Enables sharp connection points in the freehand curve.

NORMALISE BASIS

This parameter (on by default) normalizes the curve. It is applicable only for creating curves using Breakpoints and Uniform parameterization.

When creating NURBS or Bezier curves with this parameter disabled, the knot spacing will be 1.0. The same knot spacing is used when creating a curve using CV's instead of breakpoints.

When creating curves with Breakpoints and a non-Uniform parameterization, or when using the Freehand curve option, the resulting curve basis is always normalized. When using CV's, it is never normalized by the modeler.

The advantage of normalising the curve is that knot spacing stays constant, regardless of the number of breakpoints on the curve. This can be useful when curve is input into some OPs.

37 CURVECLAY OP

37.1 DESCRIPTION

The CurveClay OP is similar to the Clay OP in that you deform a spline surface not by modifying the CVs but by directly manipulating the surface. However, instead of using a point on the surface, you use one or more faces to deform that surface. Also, CurveClay does not yet support polygonal meshes.

The combination of inputs will determine the modes of transformation. For any combination of inputs, the following parameters modify the following behaviours of the OP.

37.2 PARAMETERS

FACE GROUP

Subset of faces (NURBS, Bézier, Polygons) to project, or subset of profiles to deform, depending on how many inputs are connected. Examples include:

```
0.5 1.2-3.9 5.*
```

This group can even take surfaces (possibly intermixed with profile curves) when the 2nd input is not present, indicating that all the surface's profiles must be used. Then, the example above becomes:

```
0.5 1.2-3.9 5
```

SURFACE GROUP

Subset of spline surfaces to be deformed when all three OP inputs are connected

DIVISIONS ON FACE */divs*

The number of points to evaluate on the profiles or the faces. This OP works by using a straight line approximation of the given curve to deform the surface. Thus, more segments are slower, but the result looks better. Fewer divisions need to be specified when deforming profiles and when the rest and deforming faces have an equal number of breakpoints.

SHARPNESS */sharp*

Specifies the area around the face to deform. The larger the sharpness is, the smaller the deformation area around them (thus having a sharper pull on the surface.)

REFINEMENT */refine*

Usually, CurveClay automatically refines the surface. However, you may specify some degree of refinement control. In general, the more refined the surface is, the

smoother the result. Better refinement results in a denser surface. You should experiment with values between -1 and 1. When the value is negative, the OP will first refine the surface to the same detail level as when refinement is 0, and then it unrefines it. The lower the value, the more unrefined it becomes.

37.3 PROJECTION / DISPLACEMENT PAGES

The combination of inputs determines the way the OP deforms the surface. There are three valid combination types: 1 input, 2 inputs, and 3 inputs.

ONE INPUT

When only the surface is specified, it must contain at least one profile in order for Curveclay to work. In this case, you must specify how the surface should be deformed in the region near the profile (whether to deform the profile along a vector [X, Y, Z or User Defined], or along the surface normal). You must also specify how far out the surface should be pulled, using the *Distance* parameter.

When one or more profiles form a closed loop, you have the option to displace the inside of the loop, by enabling the option *Deform Inside of Loop*. Another option allows you to form a closed loop by using several open profiles. The algorithm is the same as the one used in the Trim OP when not treating curves individually. This might be useful for putting a regular dent into a surface (e.g. A circle, or a font). For an example, see: *\$HD/SOPs/CurveClay/CurveClaySOpeg1.hip* .

TWO INPUTS

When the surface and a set of deforming faces is specified (the deformed faces are the third input) the profiles on the surface will be snapped to the deforming faces. This option has no additional inputs. This might be useful when you want to snap any profile to any free floating curve. For an example, see: *\$HD/SOPs/CurveClay/CurveClaySOpeg2.hip* .

THREE INPUTS

When all three inputs are specified, the Rest faces will first be projected to the surface. The projection is done in two ways: along a specific vector (X, Y, Z, or User Defined) or along the *Minimum Distance* to the surface. Each point on the curve will first be projected to the surface. Then this curve will *not* be snapped to the deforming face. Rather, the curve will be deformed by the difference between the rest face and the deforming face. This option provides flexibility on all inputs. For example, this would be ideal for creating random mountain ranges.

PROJECTION SUB-PAGE

Controls curve projection. Enabled if all 3 inputs exist.

projection axis */projdir*

Choice of several projection axes:

X, Y, Z Cartesian axis - X, Y, or Z.

Minimum Distance Project curve points to their closest places on surface.

User Defined Enter the vector into the field below.

DISPLACEMENT SUB-PAGE

How to deform surface. Enabled if only 1 input exists.

projection axis */deformdir*

Choice of several projection axes:

Surface Normal The profiles will be deformed along the surface normal.

X, Y, Z Cartesian axis - X, Y, or Z.

User Defined Enter the vector into the field below.

distance */deformlen*

Distance deformed along the vector.

deform inside

Check if the inside of closed loops should be deformed.

individual

Check if multiple curves form a closed loop.

37.4 NOTES

When using CurveClay on a wrapped surface, here are some points to remember:

- You may want to use a higher number of divisions, since when going across the seam of the wrapped surface, straight line approximation would be turned off and more sample points may be needed.
- If you're not going across the seam, then the refinement of the surface is only local. However, when the deformation area of the surface gets near the seam, the refinement is done over the whole surface. Do not be alarmed if the whole surface has been refined.

37.5 SEE ALSO

- *Clay OP* p. 498

38 CURVESECT OP

38.1 DESCRIPTION

Finds the intersections or the points of minimum distance between two or more faces (polygons, Béziers, and NURBS curves) or between faces and a polygonal or spline surface.

38.2 PARAMETERS

FACE GROUP

A subset of faces (NURBS, Bézier, polygons) to act upon. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

CUTTER GROUP

A subset of faces or spline surfaces to intersect with. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

FIND ALL INTERSECTIONS

Compute intersection points if the faces touch the cutter primitive. If the button is unchecked, only the point of minimum distance will be found. Currently, finding the minimum distance between a face and a surface is not available.

TOLERANCE */tolerance*

Determines the precision of the intersection.

CUT PARAMETERS

This option will cut the faces where they intersect or at the points of minimum distance.

left-face pieces

Choose what parts of the left faces to keep:

Keep All All the face segments generated by the intersection(s)

Keep Odd-numbered Ones The odd face segments: 1, 3, 5...

Keep Even-numbered Ones The even face segments: 0, 2, 4...

Keep None Delete the entire face

right-face pieces

Choose what parts of the right faces to create:

- Keep All* All the face segments generated by the intersection(s)
- Keep Odd-numbered Ones* The odd face segments: 1, 3, 5...
- Keep Even-numbered Ones* The even face segments: 0, 2, 4...
- Keep None* No face segment

EXTRACT PARAMETERS

This option extracts intersection points or isoparms.

affect

Choose which input to operate on:

- Left Input* Extract from the left input faces
- Right Input* Extract from the right input (face or surface)
- Both Inputs* Extract from both left and right inputs

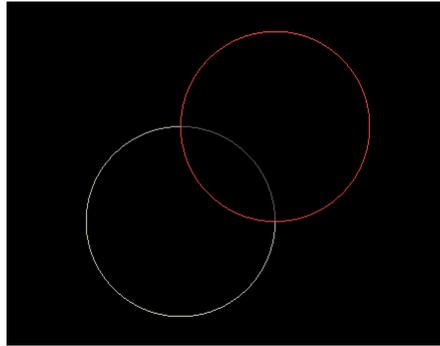
extract point

If the right input is a surface, choose between point and isoparm extraction; only points are extracted if the right input is a face

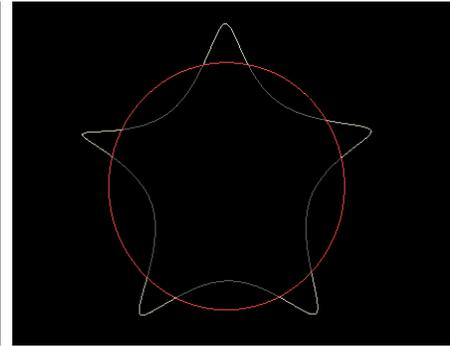
keep original

Preserve the left input faces or delete them.

38.3 EXAMPLES



Two NURBS Circles – Cut (Odd)



Modeled Star and Circle – Cut (Even)

CIRCLES

1. Place two NURBS Circle OPs, one of them slightly offset in X and Y (not Z).
2. Pipe their outputs into a Curvesect OP, make it the display OP.
3. Method: Cut. For the *Left Face*, select *Keep Odd-numbered Ones*.
4. You can extract the inside (even) or outside (odd) portion of the curve where they intersect.

STAR

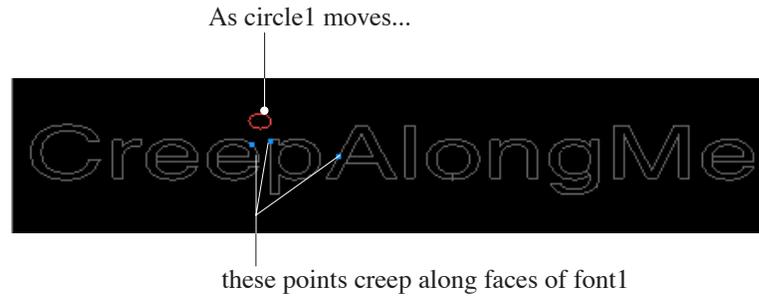
1. Model a closed NURBS curve in the shape of a star.
2. Place a NURBS Circle OP.
3. Pipe them into a Curvesect OP, make it the display OP.
4. It produces five independent NURBS curves based on their intersection with the NURBS circle.
5. You could append a Join OP to reconnect the individual NURBS curves into a new continuous curve. It smoothly connects the remaining segments.

CREEPING BEHAVIOUR ALONG A PATH

1. Place a new Font OP and then a Circle OP. Circle OP – Type: NURBS, Radius 0.1, 0.1; Centre -3, 0.3, 0.2. Font OP – Text: “Creep Along Me”.
2. Append a Curvesect OP with the Font and Circle OPs feeding into the first and second inputs respectively. Change the *Tolerance* to 1.0, and enable the *Extract* operation. Turn off *Find All Intersections*. Make it the display OP.
3. Enable the template flags on the Font and Circle OPs (use **Shift** click).
4. Home the view, and enable the *Point* display in the Viewport options.

5. As you change the Circle's X value from -3.0 to 3.0, you will notice that a number of points will creep along the paths of the curves within the text. Using a Copy OP to copy geometry to each of these points gives you a simple way of creating an irregular "flocking behaviour" along predetermined paths.

This works, because as the circle moves above the curves in the text, the points in the curves that are closest to the circle move also.



6. You can model a curve instead of using a Font OP to model several NURBS curves which are the paths to follow for more precise control over the motion.

39 DEFORM OP

39.1 DESCRIPTION

Deform works in conjunction with a Capture or Capture Proximity operation and multiple Capture Regions. The Capture[Proximity] operation assigns capture weights to points on a geometry. Each of these weights refer to a Capture Region operation. As the Capture Region moves, Deform displaces the points on the geometry according to the point weights.

Typically one or more Capture Regions are placed in an object (usually a bone) and named "capture_region". The object is then read into a Capture operation through the object's hierarchy. The object is moved to deform the Capture operation's geometry with a Deform operation.

39.2 PARAMETERS

GROUP

Optional point and/or primitive group to be deformed. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

DELETE CAPTURE ATTRIBUTES

The point capture attributes can significantly increase the memory usage of the geometry. This option will delete the point capture attributes after it deforms the geometry in order to save memory for any subsequent OPs.

DELETE POINT COLORS

You may find that you are using point coloring from the Capture OP to assist in the capturing process. If you do not need these point colors after the deform OP, you can turn this parameter on to delete the colors.

DEFORM NORMALS

Deforms the point normals to match the deformation of the points.

NORMALIZE WEIGHTS

This option specifies whether the weights used for deformation will be normalized so that they always sum to 1 for each captured point.

Turning off this option allows more flexibility in specifying capture weights, especially from imported data.

FAST DEFORM

This accelerates the deformation speed when the input object is not changing.

39.3 SEE ALSO

- *Capture OP* p. 472
- *Capture Edit OP* p. 476
- *Capture Proximity OP* p. 484
- *Objects > Character Tools* p. 376

40 DELETE OP

40.1 DESCRIPTION

Deletes input geometry as selected by a group specification or a geometry selection by using either of the three selection options: by entity number, by a bounding volume, and by entity (primitive/point) normals. You can choose to delete the selected or the non-selected geometry.

Tip: A better tool to delete points and edges is the *Edge Blast OP* p. 556.

40.2 PARAMETERS

GROUP

Specify the group or groups from the incoming OP to use as the selected geometry. If no group is specified, you must generate a selection using either the *Number*, *Bounding*, or *Normal* selection options. If given a group containing profiles, the OP deletes the profiles and leaves the parent surface intact. If the parent surface is listed in that group on its own as well, both the surface and all its profiles will be deleted.

You can specify profile curves within the group by providing a profile pattern (e.g. *.3 specifies the fourth profile in all spline surfaces).

OPERATION

Choose to delete the selected geometry or delete the non-selected geometry.

ENTITY

Select from this pop-up menu what you wish to delete.

Primitives Select primitives for deletion.

Points Select points for deletion.

GEOMETRY TYPE

Select the geometry type to delete from the selected primitives. The selection will only pertain to the geometry type specified. e.g. If you only wanted to delete polygons.

- All Types – all geometry will be selected
- Bézier Curve
- Bézier Surface
- Circle
- Mesh
- Meta-ball
- NURBS Curve

- NURBS Surface
- Particles
- Polygon
- Sphere
- Tube

NUMBER

Allows selection of deletion of entities by number. When checked, the options relative to this selection option are displayed.

number enable

When the *Enable* button is checked under the *Number* button, the selection options become active and can be used to select primitives. The fields available are listed below.

operation

When the *Number Enable* button is checked, this option deletes primitives based on a defined *Pattern* or by a *Range*.

Delete by Pattern select a pattern in the *Pattern* field below.

Delete by Range select a *Range* using the *Start/End* and *Select_of_* fields below.

Delete by Expression Delete a range using the *Filter Expression* field below.

filter expression

The *Filter Expression* provided is evaluated for every point/primitive. Wherever it is true, the entity is deleted. All the local variables of point and primitive are present, though only accessible when the right type of entities are being deleted.

pattern

Activated when *Operation* is set to *Delete by Pattern*. In this field, enter the range of primitives and profile curves (also known as curves on surface) to select. For example:

1-20 34 36 38 45-75	
0-100:2	selects every other number from 0 to 100
0-10:2,3	selects every two of three
0.0-6	selects the first six profiles on primitive 0

start/end */rangeend*

Activated when *Operation* is set to *Delete by Range*. Select the start and end of the primitive/point number selection.

select _ of _ */select1 /select2*

Activated when Operation is set to *Delete by Range*. Select every *n*th occurrence of every *m*th entity in the above Start/End range. For example:

enter: 1 and 2 selects 1 out of every 2 primitives

BOUNDING

This option is used for selecting entities based on bounding volumes: Bounding Box, or Bounding Sphere. When checked, the options relative to this selection option are displayed.

bounding enable

When the Enable button is checked under the Bounding button, the selection options become active and can be used to select entities. The bounding volume can be seen in the viewport as guide geometry.

bounding type

Selects the type of bounding volume to use:

Bounding Box entities contained within the box are selected.

Bounding Sphere entities contained within the sphere are selected.

size */sizex /sizey /sizez*

Dimensions of either the Bounding Box or Bounding Sphere in X, Y and Z.

center */tx /ty /tz*

The X, Y, and Z coordinates of the center of the bounding volume.

NORMAL

This option is used for selecting entities based on the angle of the entity normals. When checked, the options relative to this selection option are displayed.

normal enable

When the *Enable* button is checked under the *Normal* button, the selection options become active and can be used to select entities. The fields available are listed below.

The primary axis and the spread angle from the defined axis define a range of angles. If any entity normal lies within this range, then the associated entity is selected.

For example; if you want to select the polygons that are very steep in a polygon mountain terrain on the XZ axis. You would set the Direction to be 0, 1, 0 and the spread angle to around 75°. This selects all the polygons with normals that lie flat to fairly sloped. You will have deleted all of the polygons that lie flat up to polys that are at a 75° angle from the axis. You are left with all of the polygons that are 76° or greater.

direction */dirx /diry /dirz*

The default values of 0, 1, 0 create a normal vector straight up in Y, which is perpendicular to the XZ plane, which becomes the primary axis. The 1, 0, 0 points the normal in positive X, giving a normal axis perpendicular to the YZ. The plane may be positioned at an angle by using values typed in (1, 1, 0 gives a 45° angle plane) or interactively by using the direction vector jack. Use only values between 0 and 1.

spread angle */angle*

The value entered in this field generates an angle of deviation from the primary axis. This can be visualized as a cone where the radius of the base of the cone is defined by the Spread Angle and the axis of the cone is determined by the Direction axis. Viewing the primitive normals in the viewport, you can see that any primitives with normals that have an angle that lies in the range of angles defined by the cone will be selected and deleted.

backface from

This menu allows you to select an object. Typically, a camera object would be chosen. The primitives which are backface when viewed from the object specified will be grouped or deleted.

DELETE UNUSED GROUPS

If any group has 0 entries and if this parameter is enabled, then those groups are removed. If you want to keep empty groups, disable this parameter.

40.3 INPUTS / GEOMETRY TYPES

Accepts all geometry types.

40.4 LOCAL VARIABLES

You can use all local variables of the Point OP – see *Local Variables* p. 671. Here is a subset:

<i>N</i>	The number of incoming primitives / points.
<i>PT</i>	The point number
<i>NPT</i>	The total number of points
<i>TX, TY, TZ</i>	The point position

Note: For this OP, only the *Filter Expression* parameter actually supports local variables other than *N*.

40.5 USES / WORKS IN RELATION WITH

- Deleting any unnecessary geometry.
- Culling primitives from very dense geometry in order to speed up the cooking of OPs further down the chain.

40.6 SEE ALSO

- *Edge Blast OP* p. 556
- *Culling Page* p. 132 of the *Interface* section.
- *Group OP* p. 592 > Delete Groups option.

41 DISSOLVE OP

41.1 DESCRIPTION

Dissolve deletes points, primitives and edges from the input geometry and repairs the holes left behind. You can also specify point references (e.g. "p0" will delete all the edges that connect to point 0).

There is an option to recompute normals after the deletion.

ENTITY SPECIFICATION

All the selections are treated as a set of edges, and the edges that are not part of any entity in the selection do not get deleted. You can specify the entities you want to delete in the Entities field. The entities are specified as follows, each separated by a space:

<i>n</i>	<i>n</i> is an Integer	Deletes primitive <i>n</i> .
<i>pa</i>	<i>a</i> is an Integer	Deletes point <i>a</i> .
<i>nem</i>	<i>n</i> and <i>m</i> are Integers	Deletes edge <i>m</i> of primitive <i>n</i> .
<i>pa-b</i>	<i>a</i> and <i>b</i> are Integers	Deletes edge between points <i>a</i> and <i>b</i> .

41.2 PARAMETERS

GROUP

The points, primitives or edges to delete.

OPERATION */invertsel*

Determines whether the edges specified are to be deleted, or the only ones kept.

RE-COMPUTE NORMALS */compnorms*

Calculates the normals on remaining geometry after the deletion occurs.

41.3 SEE ALSO

- *Delete OP* p. 543
- *Edge Blast OP* p. 556

42 DIVIDE OP

42.1 DESCRIPTION

This OP divides incoming polygonal geometry. It will smooth input polygons, dividing polygons, as well as sub-divide input polygons using the *Bricker* option. Bricker is also useful for polygon faces with more than four edges to chop them up into quads and triangles allowing for proper shading when using deformation tools.

42.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

CONVEX POLYGONS

When checked, this option will convex all incoming polygons to the maximum number of sides as specified in the field below. This is useful for reducing the number of sides of polygons that are concave and not shading properly, e.g. to generate triangles from all incoming geometry, check this option and change the *Maximum Number of Sides* field below to 3. You will see that all of the polygons are reduced to three sides maximum (triangles).

MAXIMUM EDGES */numsides*

Input value determines the maximum number of edges that all of the input polygons will be reduced to if they exceed this number. Sets the maximum number of sides for convexed polygons. There must be at least three sides. Using small numbers in the range of 3-6 gives the best results; the resulting polygons are not so long and thin. If input polygons have fewer edges than specified, no change will be executed on those polygons.

SMOOTH POLYGONS

If checked, this feature will divide the polygons which are adjacent to each other and are not flat, as in the corners of a box. The threshold of the smoothing and the amount of polygon divisions that are added are controlled by the fields below.

All incoming geometry must have shared points for Smooth Polygons to work. You may have to pass the geometry through a *Facet OP > Consolidate Points* to achieve this. For example, to create a dice from a box, add a *Box OP* followed by a *Facet OP > Consolidate Points* then through the *Divide OP* with *Smooth Polygons* enabled; set the *Weight* fields set to: 4, 0.5 and the *Divisions* set to: 2.

WEIGHT */weight1 /weight2*

Determines the localization effect of the added polygons. You can isolate the divisions to where there are edges in the geometry with values greater than 1 enhancing the edges by smoothing out the angle transitions. Values less than 1 tend to put the divisions in the flatter areas of the source geometry and alter the shape of the geometry considerably by pulling in the edges.

DIVISIONS */divs*

Determine the level of sub-divisions for the Smooth Polygons option. A value of 1 doubles the number of polygons at the corners, a value of 2 will add twice as much sub-division. Values of 3 and greater may add a substantial number of polygons and should be used with care – you can exponentially increase the complexity.

BRICKER POLYGONS

Selecting this option divides the input polygon geometry into grid-like squares, though the output is not a mesh. Bricking creates new polygons. It can be used to divide a surface so that it deforms more naturally when wrapped on another surface using Creep, or twisted with a Lattice OP.

It can also be used to divide a planar surface into smaller pieces to apply point colors using a Point OP. The size and location of the square divisions created by bricking is determined by the following three options.

SIZE */sizex /sizey /sizez*

Sets the size of the bricker grid divisions in each of the three axes.

OFFSET */offsetx /offsety /offsetz*

Sets the XYZ offset of the grid divisions to the Source geometry (the Bricking Grid is moved).

ANGLE */anglex /angley /anglez*

Determines the angle relative to XYZ axes, at which bricker polygons are created.

REMOVE SHARED EDGES

Eliminates any common edges.

COMPUTE DUAL

Convert the polyhedron into its point/face dual.

43 DUPLICATE OP

Duplicate lets you make copies of a subset of input primitives and apply transformations to these copies.

The Duplicate OP is different from the Copy OP in the following ways:

- Copying onto a template is not supported
- Copies are appended to the source geometry, not extracted
- Duplicate is much faster than Copy because it does less work

43.1 PARAMETERS

SOURCE GROUP

A subset of input primitives to copy from.

NUMBER OF COPIES */ncy*

The number of copies to be made.

TRANSFORM CUMULATIVE

Each copy is transformed relative to the previous.

TRANSFORM ORDER

Order transformations occur.

ROTATE ORDER

Order rotations occur.

TRANSLATE */tx /ty /tz*

Translation along xyz axes.

ROTATE */rx /ry /rz*

Rotation about xyz axes.

SCALE */sx /sy /sz*

Non-uniform scaling along xyz axes.

PIVOT */px /py /pz*

The local pivot point of the copy.

UNIFORM SCALE */scale*

Uniform scaling.

PRESERVE NORMAL LENGTH

Normals to maintain length after transformations.

CREATE OUTPUT GROUPS

Creates a primitive group and places all copies in it.

COPY GROUPS

The name of the group to create.

43.2 LOCAL VARIABLES

CY The copy number.

NCY The number of copies.

The following variables refer to the input geometry:

CEX, CEY, CEZ The centroid of the entire input geometry

GCX, GCY, GCZ The centroid of the input group

XMIN, XMAX The X extents of the bounding box of the geometry

YMIN, YMAX The Y extents of the bounding box of the geometry

ZMIN, ZMAX The Z extents of the bounding box of the geometry

SIZEX, SIZEY, SIZEZ The size of the bounding box

BBX, BBY, BBZ The point's relative position in the bounding box

43.3 SEE ALSO

- *Copy OP* p. 519
- *AttribCopy OP* p. 440

44 EDIT OP

44.1 DESCRIPTION

The Edit OP is much like the Transform OP, and allows interactive editing of the source geometry. Simply click and drag points as desired, using the **Q** key to quit editing one point and start another.

Edit is a cumulative operation, meaning that several editing tasks can be performed within a single operation. Thus, repeated uses of Edit will not create an Edit node in the network each time a new editing operation begins or the selection changes.

This makes this operation best for quick, repeated edits, since only one Edit node will be created in the network. This differs from most other operations, like Transform, that create new nodes for each new operation.

Use of the Edit operation will result in cleaner, more compact networks, in addition to improved overall performance. On the other hand, Edit operations are not animatable, and past edits cannot be modified procedurally.

To maintain a history of operations, or to animate operations, use Transform, Soft Transform, Peak, or Soft Peak instead.

COMPARE EDIT WITH TRANSFORM

Edit OP	Transform OP
Unlimited operations within a single node.	One operation per node.
Not animatable.	Animatable.
Cannot change history of operations.	Can change history in a chain of operations.

44.2 KEYBOARD SHORTCUTS

Note: See Handles p. 408 for a description of the available manipulator Handles.

- Q** Quit tweaking the current point, and start another (if *Secure Selection* isn't turned off).
- T** Toggle Translation Handle.
- R** Toggle Rotation Handle.
- E** Toggle Scale Handle.
- Y** Toggle Peak Handle.

Note: If you want to quickly modify geometry by repeatedly clicking and dragging freely. For this type of work, make sure the *Secure Selection* button on the left is turned *Off*. This will allow you to avoid pressing **Q** to start a new operation.

44.3 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

GROUP TYPE

Enter the type of elements to be referenced in the *Group* field. You can specify primitives, points, breakpoints or edges. What you enter here supplies a hint to the OP as to what kind of *Group* is being specified in order to parse it more quickly.

TRANSFORM ORDER

Sets the overall transform order for the transformations. The transform order determines the order in which transformations take place. Depending on the order, you can achieve different results using the exact same values. Choose the appropriate order from the menu.

TRANSLATE */tx /ty /tz*

These three fields move the Source geometry in the three axes.

ROTATE */rx /ry /rz*

These three fields rotate the Source geometry in the three axes.

SCALE */sx /sy /sz*

These three fields scale the input geometry in the three axes.

PIVOT */px /py /pz*

The pivot point for the transformations (not the same as the pivot point in the pivot channels). The pivot point edit fields allows you to define the point about which geometry scales and rotates. Altering the pivot point produces different results depending on the transformation performed on the object.

Tip: In order to transform from the Bounding Box centre, simply enter the variables \$CEX \$CEY \$CEZ into the Pivot parameter.

UNIFORM SCALE

Uniform scale allows you to shrink or enlarge geometry along all three axes simultaneously.

PIVOT ABOUT GROUP CENTROID

Performs the pivot around the centre of the all the elements in the group combined.

RECOMPUTE POINT NORMALS

Recomputes point normals if they exist.

PRESERVE NORMAL LENGTH

Normals to maintain length after transformations.

44.4 SEE ALSO

- *Handles* p. 408
- *Transform OP* - p. 805
- *Soft Transform OP* p. 760
- *Peak OP* p. 666
- *Soft Peak OP* p. 758
- *Sculpt OP* p. 735
- *Point OP* p. 667

45 EDGE BLAST OP

45.1 DESCRIPTION

Edge Blast deletes primitives, points, edges or breakpoints. Unlike Dissolve, it doesn't fix the hole it creates.

45.2 PARAMETERS

GROUP

Primitives, points, edges or breakpoints to delete.

GROUP TYPE

The type of entities specified in the Group field. These are the entities that will be deleted.

45.3 SEE ALSO

- *Delete OP* p. 543
- *Dissolve OP* p. 548

46 EDGE COLLAPSE OP

46.1 DESCRIPTION

The Edge Collapse operation collapses edges and primitives to their centroids.

The notation for the group field is as follows:

n	n is an Integer	Deletes primitive n .
pa	a is an Integer	Deletes point a .
nem	n and m are Integers	Deletes edge m of primitive n .
$pa-b$	a and b are Integers	Deletes edge between points a and b .

46.2 PARAMETERS

GROUP

Edges to collapse.

REMOVE DEGENERATE

Cleans up any degenerate polygons.

RECOMPUTE POINT NORMALS

Recomputes point normals if they exist.

46.3 SEE ALSO

- *Edge Blast OP* p. 556
- *Dissolve OP* p. 548
- *Fuse OP* p. 588
- *Facet OP* p. 570

47 EDGE CUSP OP

47.1 DESCRIPTION

The Edge Cusp Operation creates cusps along the given edges. In other words, it sharpens the specified edges by unquining their points and recomputing their point normals.

The two end points along the path to cusp are not unquined, therefore this operation requires at least two connected edges in order to have any effect on the geometry. With only two edges, the point that is shared between them is the only point unquined, and the cusp fades into the other two points. This provides for the ability to tailor the cusp by choosing the edges along which the cusp will fade.

47.2 PARAMETERS

GROUP

The edges with which to create a cusp.

UPDATE POINT NORMAL */updatenorms*

Recomputes the point normals if they are present in the input geometry.

47.3 SEE ALSO

- *Facet OP* p. 570

48 EDGE DIVIDE OP

48.1 DESCRIPTION

Edge Divide inserts points on the edges of polygons and optionally connects them. It allows you to specify the number of segments you want in the result, and whether or not the new points will be shared by the coincident edges.

The group field expects a string as follows, each word separated by a space:

<i>n</i>	<i>n</i> is an Integer	Deletes primitive <i>n</i> .
<i>pa</i>	<i>a</i> is an Integer	Deletes point <i>a</i> .
<i>nem</i>	<i>n</i> and <i>m</i> are Integers	Deletes edge <i>m</i> of primitive <i>n</i> .
<i>pa-b</i>	<i>a</i> and <i>b</i> are Integers	Deletes edge between points <i>a</i> and <i>b</i> .

48.2 PARAMETERS

GROUP

The edges on which to insert points.

DIVISIONS */numdivs*

The number of segments resulting from the original edge.

APPLY TO ALL COINCIDENT EDGES */applytoall*

If the edge is specified by primitive, this option toggles between whether or not the edge division will apply to all the edges that share the points. If the edge is specified by points, this option has no effect.

SHARE NEW POINTS */sharedpoints*

Toggles whether or not each coincident edge will get its own points, or if they will all share the new points.

CONNECT POINTS */connectpoints*

Toggles whether or not the new points should be connected. The points are connected in the order that the edges are specified.

CLOSE PATH */closepath*

Toggles whether or not the path traced out by the connected points should be closed. When a path is closed, the point on the last edge is connected to the point on the first edge. Since the connections are done with a minimum distance criteria, to get the desired results, the length of the path from the first edge to last edge should be smaller than the length of the rest of the path.

48.3 SEE ALSO

- *Refine OP* p. 719
- *Resample OP* p. 723

49 EDGE FLIP OP

49.1 DESCRIPTION

This Operation flips edges from input geometry. You can set the number of times to flip the edge (*Number of Cycles*). You can also choose to either cycle vertex attributes or not. This is useful for when vertices have texture coordinates.

49.2 PARAMETERS

GROUP

The edges to flip.

CYCLES */cycles*

The number of times to flip the edges.

CYCLE VERTEX ATTRIBUTES */cycleattribs*

Toggles whether or not the vertex attributes cycle with the edges or remain fixed.

50 ENDS OP

50.1 DESCRIPTION

The Ends Operation allows you to close, open and clamp primitive end points. It provides the same functionality as from Face/Hull page in Primitive SOP.

50.2 PARAMETERS

SOURCE GROUP

Primitive and/or profile group to operate on.

PRESERVE SHAPE U/V

If clamping or closing rounded, preserves shape.

CLOSE U/V

Closes, opens, or unrolls primitive in U/V direction.

CLAMP U/V

Clamps NURBS endpoints to original positions.

50.3 SEE ALSO

- *Primitive OP* p. 697

51 EXTRUDE OP

51.1 DESCRIPTION

Allows you to extrude geometry along a normal. It can be used for:

- Extruding and bevelling Text and other geometry
- Cusping the bevelled edges to get sharp edges
- Making primitives thicker or thinner.

In order to do so, it uses the normal of the surface to determine the direction of extrusion. In the case of planar or open polygons, the normal is difficult to determine, and may not always provide the result that you expect. Turn on the *Primitive Normals* display in the Viewport display options to see the normals.

You can also extrude along an arbitrary vector, or along the face normal, and you can specify the number of divisions in the extrusion.

51.2 PARAMETERS

SOURCE GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified for the source. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

X-SECTION GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified for the cross-section. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

51.3 PARAMETERS – VALUE PAGE

FUSE POINTS */doFuse*

This should almost always be turned on when cross-sections are used. It consolidates points of polygons that would otherwise cross or overlap when the bevel takes place.

<i>Clamp All Points</i>	Guarantees no overlapping points.
<i>Clamp Minimal Set</i>	Provides more uniform control with overlapping only in extreme cases.
<i>Clamp Individual Faces</i>	Implemented only for backward compatibility.

FRONT FACE

Control how the front face of the extrusion should be built. You may wish to have a “No Output” because some faces are never actually seen when doing animation and, therefore, would only take up additional overhead if left on.

<i>No Output</i>	No face is created
<i>Output Face</i>	Faces are created
<i>Convex Face</i>	Create faces built with Convex Polygons (use this option if faces are to be deformed, i.e. Twist OP, Lattice OP).

BACK FACE

This value controls whether or not the back of the extruded object will have a face or not. The options are the same as the *Front Face* options above.

SIDE MESH

Controls how the cross-section(s) will be extruded. If the input cross-section is a Bézier or NURBS curve, the surface will be constructed with a patch of the same geometry type.

• <i>No Output</i>	No mesh is created.
• <i>Rows</i>	Creates horizontal lines.
• <i>Columns</i>	Creates vertical lines.
• <i>Rows & Cols</i>	Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon). Compare them in the Model Editor.
• <i>Triangles</i>	Build the grid with Triangles.
• <i>Quadrilaterals</i>	Generates sides composed of quadrilaterals (default).
• <i>Alternating Triangles</i>	Generates triangles that are opposed; similar to the Triangles option.

INITIALIZE EXTRUSION

If the cross-section face that you created doesn’t match up nicely to the size of the geometry you are extruding, this command will scale and translate it so that it fits nicely.

The reason it might not be nice to begin with is that the curve wasn’t drawn exactly on the world-axis in Model-mode and/or was drawn at a grossly different scale than the object it is extruding.

THICKNESS */thickxlate /thickscale*

The first value controls the tangential offset of the cross-section profile. There is no effect if a straight (default) cross-section is used. The second value controls the x-scale of the cross-section profile. Negative scaling values are valid.

Why Two Parameters? Internally, the two Thickness parameters control the individual scale and offsets for the X and Y values of the cross-section. So if the X points = 0 (vertical) then the */thickscale* parameter doesn't have any affect.

DEPTH */depthxlate /depthscale*

The first value moves the entire extrusion forward or backwards in the direction of the extrusion. The second value controls the distance between the cross-section and the source input curves.

VERTEX */vertex*

Translates the cross-section such that the vertex specified is at the cross-section origin.

CUSP POLYGONAL SIDES

Determines whether or not sides are to be smooth-shaded or faceted using the angle value in *Side Cusp Angle* field.

Usually when rendering, the renderer will render smooth all polygons with shared vertices (default). Cusping overrides this and lets you specify at which angle between adjacent polygons, a sharp edge (faceted edge where vertices are unshared) should be displayed instead.

SIDE CUSP ANGLE */cusppangle*

When checked, this value will control the angle at which faceting of the sides will occur. A value of 20 is default.

CONSOLIDATE FACES TO MESH

If selected the extrusion will share points between the front face and the first row of the side mesh and between the last face and the last row of the side mesh.

REMOVE SHARED EDGES */removeSharedSides*

Prevents the creation of duplicate sides.

51.4 PARAMETERS – GROUPS PAGE

CREATE OUTPUT GROUPS

When this option is checked, it causes the Extrude OP to generate three new groups representing the primitives belonging to the: Front faces, Back faces, and the Side bevel/extrusion. The name of the groups are determined by the three option fields below.

Front Group Output group name to create for front face geometry.

<i>Back Group</i>	Output group name to create for back face geometry.
<i>Side Group</i>	Output group name to create for side bevel/extrude geometry.

51.5 INPUTS / GEOMETRY TYPES

SOURCE INPUT

Input the OP containing the curves that you want to extrude. The input can be polygons, Bézier curves, NURBS curves, or any combination of the three types.

CROSS-SECTION INPUT

The OP on this input will be used to define a cross-section. If not specified, a straight line is used to extrude the object. This cross-section should be an open Bézier, NURBS, or polygon drawn in the XY Plane.

51.6 USES / WORKS IN RELATION WITH

This OP is mainly used for generating bevels and extrusions of text with input cross-sections are from a Font OP. Any curve or group of curves can be used as input.

51.7 TIPS

OFFSETS

This OP can be used for generating two offsetting curves where the distance between the two curves remains constant. To do this, make sure that you set *Side Mesh* to *No Output*, the first thickness to zero and adjust the second to increase or decrease the distance of the offset.

FIXING STRAY NORMALS

If your geometry contains normals that are pointed in many directions (say after reading geometry from a File OP, or if you have a lot of open or non-planar polygons), you can fix it so that they are suitable for extrusion.

Do this by appending a Group OP to the OP that contains your geometry, enable *Normal*, and reduce the *Spread Angle* to something less than 180° degrees (e.g. 90). Then append a Primitive OP, which should work on the group made in the Group OP. In the *Face/Hull* page, set the *Vertex* menu to *Reverse*.

Now the normals in your geometry will all be oriented in the roughly the same direction, and ready for extrusion. To narrow the tolerance, decrease the *Spread Angle* further.

51.8 MOUSE AND KEYS -

<i>none</i>	Volatile key
	Drag to move the selected extrusion to a new location.
 	Moves the extrusion in direction of the height vector.
	Finish the extrusion.

51.9 DESCRIPTION -

This Operation allows you to extrude a set of faces. This is done by selecting one or more faces and entering the extrusion Operation. Once in the extrusion Operation, select a curve with  and drag it to a new location. An extrusion is automatically built between the dragged location and the original curve. This extrusion consists of two faces and a side mesh, and is very similar to the Extrude OP's operation. (If the  key is pressed while the primitive was selected by its normal, each primitive in the selection will be extruded along its normal). After the initial dragging, the current extrusion may be modified by changing any of the parameters below.

In addition to primitives, points and edges may be extruded by entering this Operation with that respective selection type enabled apriori. Extruded points create open polygons; extruded edges create meshes or closed polygons.

51.10 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog, see *Operation Parameters* - p. 568.

FRONT

Includes the original curve in the extrusion.

SIDE

Includes a side mesh in the extrusion between the front and back curve locations.

BACK

Includes a final curve in the extrusion which serves as a cap.

INSET

This value defines an approximate distance to offset the curve during the extrusion.

51.11 EXTRUDE OPERATION MENU -

FRONT FACE TOGGLE

Toggles the creation of the extrusion's front face.

BACK FACE TOGGLE

Toggles the creation of the extrusion's back face or cap.

SIDE MESH TOGGLE

Toggles the creation of the extrusion's side meshes.

FINISH EXTRUDING

Completes the extrusion of the geometry.

51.12 OPERATION PARAMETERS -

DISTANCE

Along with Direction, this defines the final position of the extruded curve in terms of distance travelled from the original selection.

SCALE

These values scale the extruded curve along its local axes. In this situation, the X axis is considered to be parallel to the first edge of the curve and the Y axis is perpendicular to the X axis and the curve normal.

DIVISIONS

This defines the number of rows when Side output is selected.

DIRECTION

Direction defines the trajectory of the extrusion's movement away from the original curve.

FUSE POINTS

This handles cases where non-zero offsets produce self-intersecting extrusions, by clipping the geometry to a common point.

CONSOLIDATE FACES TO MESH

If selected, the extrusion will share points between the front face and the first row of the side mesh and between the last face and the last row of the side mesh.

CONVERT MESH TO POLYGONS

If selected, will convert any side mesh to polygons.

52 FACET OP

52.1 DESCRIPTION

This OP lets you control the smoothness of faceting of a given object. It also lets you consolidate points or surface normals.

Facet, like Divide, works as a pipeline to change geometry in stages. For this reason, Compute Normals appears twice. For example, you can compute surface normals before making vertices (the points of each polygon) unique, which gives you the unusual result of smooth shading and unique point, as the normals get computed while the points are still shared.

Tip: This OP is great for cleaning up geometry read in from *.dxf* files. It corrects flipped normals present in some *.dxf* files.

52.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

MAKE NORMALS UNIT LENGTH

Checking this option will normalize the length of normals to a length of one unit.

COMPUTE NORMALS

Checking off this option means that the surface normals will be computed. Where points are shared between polygons, smooth shading results, and where points are not shared (unique), faceted edges result. If you compute normals at this stage, they are computed based on the incoming geometry.

UNIQUE POINTS

Makes each vertex have a unique point. The result of selecting this option is that all vertices are made into unique points, thus making all edges hard, with no smooth shading.

CONSOLIDATE

Consolidate eliminates the redundancy of having many points that are close to each other, by merging them together to form a fewer set of common points. Consolidate is useful for cleaning up an edge that may appear between adjacent polygons that have been merged.

No Consolidate

No consolidation.

Consolidate Points Slow / Fast

Consolidates points that lie within a certain distance of each other to a single point. Set the distance with the *Distance* parameter.

Fast sacrifices some accuracy for speed increases.

Consolidate Normals Slow / Fast

Averages the normals of points that lie within the specified distance of each other, so that the shading appears as if the points have been consolidated without actually consolidating them.

Fast sacrifices some accuracy for speed increases.

DISTANCE

/dist

Points and normals less than this distance apart will be consolidated, or have their normals averaged, based on the setting in the *Consolidate* menu.

Usually very small numbers, such as 0.01 should be used here. If the value is 0.0, then points must be in exactly the same position (co-incident) in order to be considered for the consolidation/averaging function.

ORIENT POLYGONS

Orients all polygons so they have the same winding direction.

CUSP POLYGONS

Most of the time, you want some polygons to be smooth shaded and others to be faceted. Usually polygons that meet at low angles should be smooth shaded, and polygon edges that meet at sharper angles should be faceted.

CUSP ANGLE

/angle

Cusping allows you to specify the threshold angle at which the edges become faceted. A good typical value is 20°.

REMOVE DEGENERATE

Sometimes (not often) your geometry can get messed up, where there are points hanging around that are not used for anything, or there are primitives that don't make sense. This option checks for these cases and removes them.

COMPUTE NORMALS

Again, allows you to compute the normals after the consolidation or cusping stages. You should select this if you have set either the *Cusp* or *Consolidate* option.

52.3 INPUTS / GEOMETRY TYPES

Accepts all geometry types.

52.4 LOCAL VARIABLES

none.

53 FILE OP

53.1 DESCRIPTION

This OP allows you to read a geometry file that may have been previously created in the Model Editor, output geometry from a OP, or data generated from other sources.

53.2 PARAMETERS

GEOMETRY FILE

Contains the full pathname of the geometry file to be read in.

Some of the geometry formats that can be read:

<i>PRISMS</i>	<i>.geo, .bgeo, .bpoly, .poly</i>
AutoDesk	<i>.dxf</i>
Inventor	<i>.iv</i>
Wavefront	<i>.obj</i>
Alias Render Format	<i>.sdl</i>
Adobe Illustrator *	<i>.eps</i>

Other formats can also be read. See *Geometry Formats Supported by Houdini* p. 253 of the *Formats* section for a complete listing.

* *Adobe Illustrator Files (up to version 5.5) can be loaded when saved as an .eps file. This will correctly map the outline paths to flat 3D geometry. Colors, fills, and patterns, however, are not supported.*

53.3 USES / WORKS IN RELATION WITH

Used to input geometry from disk to other modifier OPs.

53.4 SEE ALSO

- *Model OP* p. 645
- *Geometry Formats Supported by Houdini* p. 253, *Formats* section.

54 FILLET OP

54.1 DESCRIPTION

The Fillet OP is used to create smooth bridging geometry between two curves / polygons or two surfaces / meshes.

Filleting creates a new primitive between each input pair and never affects the original shapes. This is in contrast to the Join and Stitch OPs. The Join OP converts and possibly changes the connected ends of primitives, and stitching changes the original shapes but does not change the number of resulting primitives.

Please refer to the *Align OP* p. 430 for a discussion of “left” and “right” primitives as well as the option of an auxiliary input.

■ **Note:** Trim curves are not taken into account by a fillet. To do this, use the Join OP.

54.2 PARAMETERS

GROUP

Which primitives to fillet. If blank, it fillets the entire input. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

FILLET

Can optionally fillet subgroups of n primitives or every n th primitive in a cyclical manner. Example: Assume there are six primitives numbered for 0 - 5, and $N = 2$. Then:

- a) Groups will fillet 0-1 2-3 4-5
- b) Skipping will fillet 0-2-6 and 1-3-5.

N

Determines the number of primitives to be either grouped or skipped. $N \geq 2$.

WRAP LAST TO FIRST

Connects the beginning of the first primitive to the end of the last primitive filleted, or, if only one primitive exists, it creates a fillet between its ends.

DIRECTION

This menu determines the parametric direction of the filleting operation, which can be in U or V, and is meaningful only when the inputs are surfaces. The U direction is associated with columns; the V direction refers to rows.

LR SCALE *//rscale1 //rscale2*

Use to control the direction and scale of the first and last segments of the fillet.

LR OFFSET *//roffset1 //roffset2*

Controls the position of the first and last segments of the fillet.

MATCH INPUT TO FILLETS

If selected, then the inputs are modified in such a way that the isoparms appear continuous from one primitive, through the fillet to the other primitive. Also, the primitives are promoted to the same type and order. This will minimize if not eliminate any artifacts introduced in rendering at the cost of more refined geometry.

CUT PRIMITIVES

If selected, the primitives are trimmed at the point the fillet begins.

54.3 SEE ALSO

- *Join OP* - p. 602
- *Stitch OP* p. 775.

Triangles option.

U ORDER */orderu*

If the input is a face, this is the order of the Spline curve to be generated. If the input is a surface, this is the order of the fitted spline surface in the U parametric direction.

V ORDER */orderv*

If the input is a surface, this is the order of the fitted Spline surface in the V parametric direction. The V order is irrelevant if the input is a face.

55.3 PARAMETERS – APPROXIMATION PAGE

Approximation fitting is used primarily to generate a lean, smooth shape from a dense data set. The result is a primitive that approximates the positions and attributes of the data points but does not necessarily touch these points. The only points the fitted curve or surface goes through are the end-points of the data set. If the fitted primitive is required to go through all the points, fitting by interpolation is the answer.

The approximation fit is capable of producing very reasonable shapes with far fewer control vertices than the number of data points. Although the result is unlikely to match the original shape identically, it can come very close to it, depending on the parameters it is set to. For this reason, approximation fitting is often used as a data reduction tool and performs best when the size of the data set is large.

The fitted primitives are generated open or wrapped based on the “open” property of the inputs. For best results, the input primitives should be open.

TOLERANCE */tol*

This is the primary precision factor in approximation fitting. The smaller the tolerance, the closer the fit and the higher the number of generated vertices. If a small tolerance causes unwanted twists or contortions in the fitted primitive, it may help to vary the Spline order and/or enable the Multiple Knots flag.

SMOOTHNESS */smooth*

For a set tolerance, the smoothness factor allows for more or less roundness in the generated shape. If this parameter is zero, it does not mean that the fit will be sharp. It simply indicates that no additional smoothing is required past the level of smoothness already achieved with the given tolerance.

U / V MULTIPLE KNOTS

Sometimes the data set has sharp bends that must be preserved in the fitted shape. In this case, inserting multiple knots in the areas of sharp curvature will usually produce the right effect. Sometimes, however, the simulated sharpness induces unwanted twists immediately before or after the corner. Lowering the Spline order and/or reducing the tolerance may help diminish this side-effect.

55.4 PARAMETERS – INTERPOLATION PAGE

Interpolation fitting is used primarily to generate a shape that goes through (i.e. interpolates) a complete set of data points and their attributes. As opposed to approximation fitting, interpolation thrives on small data sets. Moreover, unlike the approximation method, this one does not produce a leaner structure than the input it fits. In certain cases it even generates a higher CV count than its input. For this reason, the use of interpolation fitting should be limited to those cases where point interpolation is paramount, such as building precise animation paths.

SCOPE

Scope establishes the interpolation method.

<i>Global</i>	Global interpolation take the whole data set into account at once and generates exactly as many CVs as data points is it given.
<i>Local</i>	Local interpolation takes a more geometric approach, building the curve or surface one span at a time, using only local data at each step. The local method generates more CVs than the number of data points it is given, but it usually yields a tighter fit than the global method. The local approach is also less computationally expensive than the global one, and handles cusps and local perturbations better. Local interpolation is available only for curves.
<i>Breakpoints</i>	Breakpoint interpolation is a variant of global interpolation that satisfies the requirement that the locations of data points coincide with the breakpoints of the generated curve. The breakpoints of a Spline curve are the image of the Spline basis on the curve. Breakpoint interpolation is available only for curves.

U DATA PARAMETER

Specifies the parameterization of the data in the U direction (the only direction if the input is a curve). The data parameterization can be uniform, chord length, or centripetal.

Uniform

Uniform parameterization uses equally spaced parameter values. It works best when the geometry is very regular. When the data is unevenly spaced, this approach can produce very unintuitive shapes, and is not recommended.

Chord Length

Chord length computes the parameterization of the data based on the relative distances between successive data points. It is the most commonly used approach because it tends to produce the most accurate results.

Centripetal

Centripetal parameterization is similar to chord length, but yields better results when the data has very sharp corners.

V DATA PARAMETER

V data parameterization is identical to U data parameterization, but it affects the V direction when the input is a surface. It is not used when the input is a face.

U WRAP

This menu determines whether the fitted curve should be closed, or whether the fitted surface should be wrapped in the U parametric direction. The options are to open (Off), close (On), or inherit the closure type from the input primitive.

V WRAP

This menu determines whether the fitted surface should be wrapped in the V parametric direction. The options are to open (Off), close (On), or inherit the closure type from the input primitive. *V Wrap* is ignored when the input is a face.

FIT CORNERS

Specifies whether corners in the data should be preserved when doing local curve interpolation.

56 FONT OP

56.1 DESCRIPTION

This OP allows you to create 3D text from Type 1 and TrueType (NT only) fonts. You can customise the list of fonts that are available by copying an Adobe Type 1 font into any directory, and enabling it with the Font Manager.

INSTALLING FONTS IN NT

To install Fonts, use the supplied *gfont* application (use: the *Start Font Manager* button, or run it as a standalone by typing: *gfont -i* in a shell window).

For Type 1 fonts, you should be using the .pfb files instead of the .afp & .afm files. The Font Manager will install these properly for you. In the Font Manager, click on the  button and select your font. Click on *Install Font* to add it to the list. Make sure that the *font.index* file is not read-only.

INSTALLING FONTS ON IRIX

If you have IRIX 6.2, you can use the application *AdobeTypeInstaller* to convert Macintosh or PC fonts on a CD-ROM to SGI format directly. If not, you may need to use a program like *Fetch* to copy the fonts from a Macintosh to your station. Be sure to set the transfer to: Binary/MacBinary for the transfer. Fonts not containing Macintosh headers are also supported.

You should copy fonts into the directory: */usr/lib/DPS/outline/* , and then add their location to the file located in: *\$HFS/houdini/font/font.index* . They will then be ready to be used in the Font OP after restarting Houdini. The fonts located in *\$HFS/houdini/fonts_gl* are not for the Font OP, they are for the Houdini UI only.

56.2 PARAMETERS

PRIMITIVE TYPE

Select from the following types. For information on the different types, see the *Geometry Types* section. *Bézier Curves and Polygons* provide the most efficient use of memory, because they use polygons for letters containing straight segments, and Bézier curves for all others.

- Bézier Curves and Polygons
- Béziars Only
- Polygons

Note: Due to an Open GL bug, holes in *Bézier* fonts may shade incorrectly in Gouraud mode; however, they will render correctly in both *mantra* and RenderMan.

START FONT MANAGER

Use this command to install new fonts into Houdini.

The command for starting the font manager is:

```
gfont [-s] -i (i=interactive mode), or
gfont [-s] -r fontname (for batch processing fonts)
```

USE FONT

Choose the font to create the text. By clicking on the  button a File Dialog will appear, and clicking  brings up a menu of the most used fonts, chosen by the Font Manager.

TEXT

Enter the text you want to generate here.

Your text can contain the following special characters:

<code>\</code>	Take the next character literally (so you can use the / and ` characters in your text);
<code>`string`</code>	Evaluate the <i>string</i> contained by the backquotes (above the  key) as an expression;
<code>\n</code>	Start a new line;
<code>\xxx</code>	Specify a character by it's ASCII code (e.g. \007).

Example: If you put something like `\\$F3` in the text string, you should see all the possible characters of a font as you play the animation (set the last frame to 256).

entering expressions as text

You can use expressions for the text. For example, entering:

```
'substr("hello world", 0, $F)'
```

causes the letters the eleven letters of the text to appear in succession during the first eleven frames.

reading in a text file

To read a text file into the Font OP, you can use the `system()` function. However, you will have to manually add a `\n` to the end of each line in the text file:

```
`system("cat myfile.txt")`
```

You can elaborate this to filtering the text through the UNIX `sed` command in order to add the `\n` characters to the end of each line automatically:

```
`system("sed -e 's/$/\\n/' ~/myfile.txt")`
```

which filters and displays a file called `myfile.txt` in your home (`~`) directory.

CENTER TEXT HORIZONTALLY

This check box allows you to center the text horizontally about $X = 0$.

CENTER TEXT VERTICALLY

This check box allows you to center the text vertically about $Y = 0$.

CENTER */tx /ty /tz*

Translates the geometry in x,y and z.

SCALE XY */sx /sy*

Scales the text in the X and Y axis.

KERN XY */kernx /kerny*

Letter spacing in the X direction. Line spacing in the Y direction if there are multiple lines. If you need manual character-by-character, you can do it in Model mode.

ITALIC ANGLE */italic*

Doesn't actually give an italic version of the font, but rather obliquates the text by shearing it the specified number of degrees. A negative number makes the text slant to the left.

LEVEL OF DETAIL */lod*

Adobe fonts are defined by Bézier curves. If polygons only is selected, the Font OP converts these to polygons. This value adjusts the number of points in the polygons that it gets converted to.

HOLE FACES

Generates holes in polygons and Bézier faces.

56.3 INPUTS / GEOMETRY TYPES

Bézier, Polygon.

56.4 USES / WORKS IN RELATION WITH

- *Extrude OP* p. 563

57 FORCE OP

57.1 DESCRIPTION

Adds force attributes to the input metaball field that is used by either Particle or Spring OP as attractor or repulsion force fields. In general, force values greater than 0 cause points to be attracted, less than 0 cause points to be repelled.

57.2 PARAMETERS

RADIAL FORCE

When checked, triggers a force towards or away from the centre of the metaball field, depending on the value of the force.

FORCE */radial*

When Radial Force is checked, this controls the strength of the Radial Force field.

DIRECTIONAL FORCE

When checked, enables all parameters below to allow control of specific force attributes.

DIRECTION */dirx /diry /dirz*

When Directional Force is checked, determines the direction vector axis, and activates forces along the directional vector for the directional forces below.

AXIAL FORCE */axial*

When Directional Force is checked, controls the force along the primary axis. Increasing this value will cause the particles to move up the primary axis of the metaball field as defined by the direction vector.

VORTEX FORCE */vortex*

When *Vortex Force* is checked, this field controls the amount of twist particles are given around the primary axis. Positive values cause the particles to spin clockwise, negative values cause counter-clockwise spins. It is a centrifugal force.

SPIRAL FORCE */spiral*

Controls the attraction/repulsion force perpendicular to the primary axis (Direction field). Values greater than 0 will cause the points to be drawn toward the primary axis. Values less than 0 push particles away perpendicular to the primary axis. It is a tangential force.

57.3 INPUTS / GEOMETRY TYPES

This OP accepts only metaball-type geometry.
Input of non-metaball geometry has no effect.

57.4 LOCAL VARIABLES

none.

57.5 USES / WORKS IN RELATION WITH

This OP can be used in conjunction with a Particle OP to create natural particle simulations of whirlwinds, tornadoes, water swirling down a drain, etc.

57.6 SEE ALSO

- *Metaball OP* p. 638
- *Particle OP* p. 653
- *Spring OP* p. 769

57.7 TIPS

- To view the metaball field of influence, turn on the hulls display of the input metaball/s object/s. Both the Particle and Spring OP will display these hulls as guide geometry.
- Adjust the weights of the metaball inputs to increase or decrease the field effect within the metaball field of influence.
- In general, it is a good idea to try and keep the scale of forces and force fields to a smaller rather than a larger scale. This tends to make particle movement and attraction more manageable and realistic.

58 FRACTAL OP

58.1 DESCRIPTION

This OP allows you created jagged mountain-like divisions of the input geometry. It will create random-looking deviations and sub-divisions along either a specified normal vector (the Direction XYZ fields) or the vertex normals of the input geometry. This is great for the creation of terrains and landscapes.

58.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

DIVISIONS */divs*

The number of subdivisions of the input geometry. Values in the range of 1 - 3 are reasonable to start with. Higher values cause excessive geometry and should be used with caution.

SMOOTHNESS */smooth*

The smoothness value scales the deviations. The range is usually between 0 and 1, and small numbers create larger deviations than large numbers. Smoothness and Scale have a similar effect, but there is a subtle difference. Smoothness is calculated for each iteration of the OP. The number of Iterations is, in turn set by the number of Divisions. Scale, by contrast, is a global scaling of the amplitude of the deviations, with no consideration of the stage at which they are created. Using low values of smoothness (producing large deviations) and a small scale value will give a slightly more random looking result than doing the opposite; high roughness values (small deviations) and large scale values, providing Divisions is greater than 1.

SCALE */scale*

Global setting of the fractal divisions. See the above discussion about Smoothness vs. Scale.

SEED */seed*

The random seed used for fractalising. Specifying a different integer value gives a different shape.

FIXED BOUNDARY

When enabled, this option prevents Fractal from applying any deviations to the edges (boundaries) so that you could, for example, fractalise a plane and still be able to connect the edges of the plane to the sides of the box. This is achieved by first refining the surface between all breakpoints, and then fractalized.

USE VERTEX NORMALS

Instead of using the Direction fields below, this sets the direction of the fractalisation at any given point to be the direction of the vertex normals of the input. This may be preferable when using a sphere or other rounded object as the input, as the deviations will originate from the center of the sphere instead of all being the same direction.

DIRECTION XYZ */dirx /diry /dirz*

The direction of the Fractalisation. The default values of 0, 0, 1 make the fractal deviations point in the Z direction. Can be overridden by: Use Vertex Normals.

58.3 INPUTS / GEOMETRY TYPES**SOURCE DATA**

Can fractalise polygons or meshes.

59 FUSE OP

59.1 DESCRIPTION

Fuse allows you to consolidate or unique points. You can consolidate points based on their proximity to each other (given by the distance threshold) or you can have all the points in the given group be consolidated to the first point of that group. To consolidate all points in the group to the first point, toggle the distance field off.

59.2 PARAMETERS

GROUP

Subset of points to unique or consolidate.

CONSOLIDATE TAB

Points within a specified distance of each other will share a point.

distance */dist*

Threshold distance for consolidation.

remove degenerate

Cleans up any degenerate polygons.

UNIQUE POINTS TAB

Makes each vertex a unique point.

SNAP TAB

Only point positions will be altered.

distance

Snap Type

What heuristic to use to determine the position of snapped points. It can be the average of all the points to be snapped. Or, it can snap to the least or greatest point number, which is useful if you know one of those is the real value.

Snap Distance (/tol3d)

How proximate two points have to be in order to be snapped

grid

- Grid Type (/gridtype)* How to specify the grid size.
- Grid Spacing (/gridspacing)* The number of units between each grid line.
- Grid Lines (/gridlines)* The number of grid lines every unit.
- Grid Power 2 (/gridpow2)* The same as Grid Lines, but a power of 2 is specified (i.e. $2^1 = 2$, $2^2=4$, $2^3=8$... $2^7=128$.. $2^9=512$)
- Grid Offset* A number between 0 .0 and 1.0 which specifies what offset the grid should have from an origin of (0, 0, 0).

UPDATE POINT NORMALS

Recomputes point normals, if they exist.

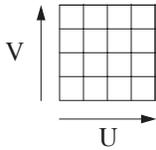
59.3 SEE ALSO

- *Facet OP* p. 570
- *UV Fuse OP* p. 836

ROWS & COLS */rows /cols*

The number of rows and columns. Rows are horizontal lines; columns are vertical lines. Two rows by two columns makes a square or rectangle. For example, one row and two columns makes a single line if *Connectivity* is set to *Rows*.

For NURBS and Bézier surfaces, the *Rows & Cols* refer to the surface hull construction, and as such, the number of Rows and Cols should not be less than the *Order* in U or V.



ORDER */orderu /orderv*

U Order Degree of spline basis +1 in the U parametric direction.

V Order Degree of spline basis +1 in the V parametric direction.

INTERPOLATION

These options are used for higher-order (i.e. NURBS, Bézier) surfaces.

End-point interpolate in U Extends the surface to touch the end point in the U direction.

End-point interpolate in V Extends the surface to touch the end point in the V direction.

60.3 INPUTS / GEOMETRY TYPES

Polygon, Mesh, NURBS, Bézier Surfaces

60.4 LOCAL VARIABLES

none.

61 GROUP OP

61.1 DESCRIPTION

This powerful OP generates groups of points or primitives according to various criteria and allows you to act upon these groups. Elements can occur in more than one group. Groups are used in many parts of the OP Editor to specify which portion/s of input geometry you wish a OP to act upon.

You can also create ordered groups. To do so, check the *Ordered* button.

61.2 PARAMETERS – CREATE PAGE

GROUP NAME

The name of the group to be created. The default name will match the OP name.

Tip: You can use expressions within the Group Name by delimiting them with back-quotes. For example, to increment the Group Name per-frame, use: `group'$F'`

ENTITY

Primitives or Points.

GEOMETRY TYPE

Select the geometry type group. The selection will only pertain to the geometry type specified. e.g. If you only wanted to group polygons.

- All Types – all geometry will be selected
- Bézier Curve
- Bézier Surface
- Circle
- Mesh
- Meta-ball
- NURBS Curve
- NURBS Surface
- Particles
- Polygon
- Sphere
- Tube

NUMBER

Allows selection of grouping of entities by number. When checked, the options relative to this selection option are displayed.

number enable

When the *Enable* button is checked under the *Number* button, the selection options become active and can be used to select entities. The fields available are listed below.

create ordered

When selected, elements in the group are traversed in the order they are selected; otherwise they are traversed in creation order.

operation

When the *Number Enable* button is checked, this option groups entities based on a defined *Pattern* or by a *Range*.

<i>Group by Pattern</i>	Select a pattern in the <i>Pattern</i> field below.
<i>Group by Range</i>	Select a <i>Range</i> using the <i>Start/End</i> and <i>Select_of_</i> fields below.
<i>Group by Expression</i>	Select a range using the <i>Filter Expression</i> field below.

pattern

Activated when *Operation* is set to *Group by Pattern*. In this field, enter the range of primitives to select. The required syntax is “S.P”, where S is the index of the parent surface, and P the profile index on that surface. You can mix primitives with profiles in the list. A mixed group is automatically ordered. For example:

0.4 2 4 2.5 3.7	selects three profiles and two primitives
1-20 34 36 38 45-75	
0-100:2	selects every other number from 0 to 100
0-10:2,3	selects every two of three
0.0-6	selects six profiles on primitive 0
0.*	selects all profiles on primitive 0
!4	selects every primitive or point except the fourth
9-0	selects first ten (in reverse if <i>ordered</i> flag is on)
!0.*	selects all profiles except those on primitive 0
*	selects all primitives or points, and no profiles

See *Scripting > Pattern Matching* p. 38 for more info on pattern matching notation.

transfer selection to pattern

This allows you to define the range of points / primitives visually by selecting them in the Viewport with the *Select Operation*. Clicking this button transfers the selected points/primitives into the *Pattern* field as a compacted range (e.g. 1-23 40 67-100). This eliminates the need for typing the point or primitive numbers manually.

Note: Point and primitive selections can be dumped directly into *Group* fields without use of the *Group OP*. Do this by selecting the points or primitives in the Viewport with the *Select Operation*. Then the pop-up ▸ menu beside the *Group* field of the *OP* you want to cook should display the selection in the input *OP* (e.g. “grid1’s Primitive Selection”). Ranges are automatically compacted.

start/end */rangeend*

Activated when Operation is set to *Group by Range*. Select the start and end of the primitive/point number selection.

select _ of _ */select1 /select2*

Activated when Operation is set to *Group by Range*. Select every *n*th occurrence of every *m*th entity in the above Start/End range. For example:

enter: 1 and 2 selects 1 out of every 2 entities

filter expression

The Filter Expression provided is evaluated for every point/primitive. Wherever it is true, the entity is added. All the local variables of point and primitive are present, though only accessible when the right type of group is being created.

BOUNDING

This option is used for selecting entities based on bounding volumes: Bounding Box, or Bounding Sphere. When checked, the options relative to this selection option are displayed.

bounding enable

When the *Enable* button is checked under the *Bounding* button, the selection options become active and can be used to select entities. The fields available are listed below. The bounding volume can be seen in the viewport as guide geometry.

bounding type

Selects the type of bounding volume to use:

Bounding Box entities contained within the box are selected.

Bounding Sphere entities contained within the sphere are selected.

size */sizex /sizey /sizez*

Dimensions of either the Bounding Box or Bounding Sphere in X, Y and Z.

center */tx /ty /tz*

The X, Y, and Z coordinates of the center of the bounding volume.

NORMAL

This option is used for selecting entities based on the angle of the entity normals. When checked, the options relative to this selection option are displayed.

normal enable

When the *Enable* button is checked under the *Normal* button, the selection options become active and can be used to select entities. The fields available are listed below.

The primary axis and the spread angle from the defined axis define a range of angles. If any entity normals lie within this range, then the associated entity is selected.

For example; if you want to select the polygons that are very steep in a polygon mountain terrain on the XZ axis. You would set the Direction to be 0, 1, 0 and the spread angle to around 75°. This selects all the polygons with normals that lie flat to fairly sloped. You will have grouped all of the polygons that lie flat up to polys that are at a 75° angle from the axis. You are left with all of the polygons that are 76° or greater.

direction */dirx /diry /dirz*

The default values of 0, 1, 0 create a normal vector straight up in Y, which is perpendicular to the XZ plane, which becomes the primary axis. The 1, 0, 0 points the normal in positive X, giving a normal axis perpendicular to the YZ. The plane may be positioned at an angle by using values typed in (1, 1, 0 gives a 45° angle plane) or interactively by using the direction vector jack. Values between 0 and 1 should be used.

spread angle */angle*

The value entered in this field generates an angle of deviation from the primary axis. This can be visualized as a cone where the radius of the base of the cone is defined by the Spread Angle and the axis of the cone is determined by the Direction axis. Viewing the primitive normals in the viewport, you can see that any primitives with normals that have an angle that lies in the range of angles defined by the cone will be selected and grouped.

backface from

This menu allows you to select an object. Typically, a camera object would be chosen. The primitives which are backface when viewed from the object specified will be grouped or selected.

EDGES */edgeangle /edgestep*

Allows you to group primitives by edges according to the following criteria:

edge angle

Specifies an angle between edges in which to group. Works only for primitive groups.

edge depth

Enter the depth of the edge (only for point groups).

point number

Enter the specific point numbers (only for point groups).

Note: *Edge Depth* and *Point #* allow for a blossoming effect from the specified point. All points that are an edge depth away from the specified point are added to the group. If there are selected points already there, the *Point #* field is disabled and the blossoming occurs from the selected points.

unshared edges (bounding polygon)

When selecting points, this option selects the points of a polygonal mesh which appear on the boundary (i.e. those which are not shared) for inclusion in the group, and orders them. In addition to polygonal meshes, this option also finds the boundaries of geo Hulls and open faces.

create boundary groups

When selecting points with the *Unshared Edges* parameter, this option becomes available. Enabling it creates new groups of the form: `<name>__<n>`, (two underscores) where the `<name>` is the Group Name specified in the *Create* page. Numbers `<n>` begin at zero, and increment as more groups are created.

These groups contain the points on *each* boundary of the surface. For example, if you have a grid with a hole in the middle of it, two new point groups are created – one containing the points for the outer boundary and one with the points from the hole. These new point groups are also ordered.

61.3 PARAMETERS – COMBINE PAGE

This page allows you to combine different groups based on specific boolean operators and combination methods.

The following operations are possible:

- Negation
- Union
- Intersect
- Exclusive Or
- Subtract

In the first field, you may enter a new group, or assign the result to an existing group.

The buttons in the center column take the negation of the group to the right of it.

61.4 PARAMETERS – EDIT PAGE

This folder allows you to edit existing groups.

CONVERT TYPE

Converts a group from a point group to a primitive group, and vice versa.

GROUP NAME

Name of the group to convert.

CONVERT NAME

New group name.

PRESERVE ORIGINAL

When checked, preserves original geometry.

RENAME

Allows you to rename an existing group to something else.

DELETE

Allows you to delete an existing point or primitive group.

61.5 INPUTS / GEOMETRY TYPES**INPUT 1 – SOURCE DATA**

The source geometry.

INPUT 2 – BOUNDING OBJECT

Bounding geometry. (Where applicable. Only works for point groups when selecting by bounding geometry.)

Accepts all geometry.

61.6 LOCAL VARIABLES

You can use all local variables of the Point OP – see *Local Variables* p. 671. Here is a subset:

<i>N</i>	The number of incoming primitives / points.
<i>PT</i>	The point number
<i>NPT</i>	The total number of points
<i>TX, TY, TZ</i>	The point position

Note: For this OP, only the *Filter Expression* parameter actually supports local variables other than *N*.

61.7 USES / WORKS IN RELATION WITH

This OP is very powerful and is ideal for combining geometry that is similar. For example, all the primitive geometry of a group of door structures for a three-dimensional architectural model could be grouped together and consequently scaled and re-sized together using a Transform OP. Works in conjunction with most filter OPs.

Group name specifiers become very powerful when you use Pattern Matching in your group specifiers, as described in: *Scripting > Pattern Matching* p. 38.

62 HOLE OP

62.1 DESCRIPTION

This OP is for making holes where faces are enclosed, even if they are not in the same plane. It can also remove existing holes from the input geometry.

The holes are made by searching for faces which are enclosed by other faces and creating bridges to the interior faces. It offers more flexibility than either the Extrude OP or Divide OP's hole-making capabilities because it can deal with interior faces which are not exactly in the same orientation as the exterior ones. It can also remove existing bridges that it finds in the input geometry if needed.

Note: This OP works with Polygonal and Bezier geometry types only. NURBS surfaces will be converted internally to Beziers.

62.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

UN-BRIDGE HOLES

This function checks for bridges to holes in the input and removes the bridges, leaving the interior freestanding. At times you may need to unhole faces so that you can connect them in some other way.

DISTANCE TOLERANCE */dist*

Interior polygons that are not in exactly the same plane as the exteriors can still become holes. The distance value tells the Hole OP how far away potential holes can be from the exterior surfaces. Faces beyond this distance will not become holes.

ANGLE */angle*

Interior faces that are rotated in relation to the exterior faces can become holes. The Angle value sets the maximum rotation of the potential holes from the exteriors. Faces beyond this rotation will not become holes.

SNAP HOLES TO OUTLINES

Points of any holes that are rotated or translated away from the exterior (or outline) plane will be moved so that they lie on the surface of the outline plane, thus avoiding twisted faces.

62.3 INPUTS / GEOMETRY TYPES

Polygons, Bézier and NURBS curves. The Hole OP can work with different types of faces simultaneously, but the resulting holed face will be made of Béziers if splines are encountered.

62.4 LOCAL VARIABLES

none.

62.5 USES / WORKS IN RELATION WITH

Building holes in geometry.

63 ISO SURFACE OP

63.1 DESCRIPTION

This OP uses implicit functions to create 3D visualizations of isometric surfaces found in Grade 12 Functions and Relations textbooks.

An implicit function is defined so that it = 0. For example with:

$$x^2 + y^2 = r^2$$

the implicit function is:

$$f(x, y) = x^2 + y^2 - r^2 = 0$$

Production Tip: The Iso surface OP can be used to pre-visualise a noise function that you are contemplating for use in a VEX shader.

63.2 PARAMETERS

IMPLICIT FUNCTION */func*

Enter the function for implicit surface building here.

e.g.1 $(\$X*\$X) / (4*4) - (\$Y*\$Y) / (3*3) + \$Z$

This formula creates a hyperbolic paraboloid, or saddle shape.

e.g.2 $(\$X*\$X) / (a*a) + (\$Y*\$Y) / (b*b) + (\$Z*\$Z) / (c*c) - 1$

This formula creates an ellipsoid.

Try loading some of the sample functions in *\$HFS/houdini/presets*

MINIMUM BOUND */minx /miny /minz*

Determines the minimum clipping plane boundary for display of iso surface.

MAXIMUM BOUND */maxx /maxy /maxz*

Determines maximum clipping plane boundary for display of iso surfaces.

DIVISIONS */divsx, /divsy, /divsz*

The density, or resolution of the iso surface polygons in X, Y and Z.

63.3 INPUTS / GEOMETRY TYPES

Creates polygons only.

63.4 LOCAL VARIABLES

X, Y, Z

Represents the variables: X, Y, and Z.

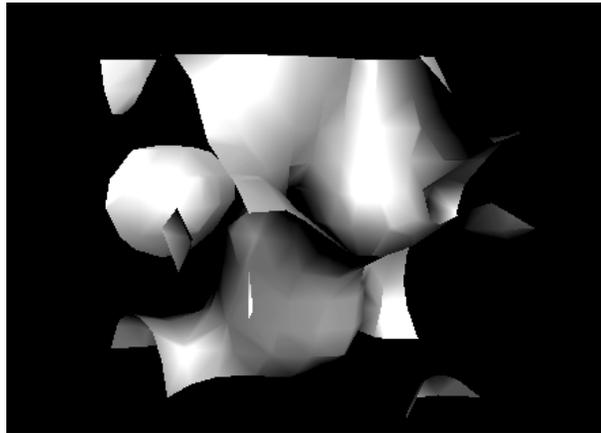
63.5 EXAMPLES

The action of the Iso surface OP is conceptually simple – it takes a user specified expression in R3 (a mathematical term meaning, “having three dimensions, each taking a Real value), and creates a surface where the function goes from being positive to being negative. In the case of the default expression ($\$X^2 + \$Y^2 + \$Z^2$), the expression is less than zero within a unit sphere, and greater than zero outside. As the OP cooks, it marches through the bounding volume specified (by default from -1 to +1 in X, Y and Z), and creates geometry where the expression equals zero.

VISUALISING A NOISE() FUNCTION

This may seem like a difficult way to define a sphere, but there’s much potential beyond this simple example using the rich array of mathematical functions (see the *Expressions* section). A simple illustration is with the *noise()* function. Try inputting the following expression:

```
noise($X, $Y, $Z)
```



INFINITE PERIODIC SURFACES

You can create an approximation of an Infinite Periodic Minimal Surface (P-Surface) with the following expression:

```
cos($X*180)+cos($Y*180)+cos($Z*180)
```

Home the view, and try increasing the *Bounds*. Bumping up the *Divisions* parameter will give you a smoother, more accurate model. However, this is computationally expensive. Although it is not as accurate, you can speed things up considerably by keeping the *Divisions* lower, and piping the output into a *Subdivide* OP.

(Courtesy: Stuart Ramsden, Vizlab)

64 JOIN OP -

64.1 DESCRIPTION

The Join OP connects a sequence of faces or surfaces into a single primitive that inherits their attributes. Faces of different types can be joined together, and so can surfaces. Mixed face-surface types are not allowed. The surfaces do not have to have the same number of rows or columns in the side being joined. Spline types of different orders and parameterization are all valid inputs. The Join OP converts simpler primitives such as polygons into Béziers and NURBS if necessary.

Joining is different from filleting (see *Fillet OP* p. 574) or stitching (see *Stitch OP* p. 775) because it takes n primitives and converts them into one after possibly changing the connected ends of the primitives. Filleting creates a new primitive between each input pair and never affects the original shapes. Stitching changes the original shapes but does not change the number of resulting primitives.

64.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

BLEND

Determines the way the primitives are joined. A blended face or surface will typically reposition the ends to be joined and convert them into a single, common point, row or column respectively. The amount of change can be reduced or eliminated by lowering the tolerance (see *Tolerance* below). When not blended, the original shapes remain unaffected. Instead, the chosen ends are joined by an arc-like fillet. In either case, the result is a single primitive.

TOLERANCE

/tolerance

The meaning of the tolerance varies with the type of join. The shapes of blended primitives will change less if the tolerance is low. If the tolerance is < 1 , a new point, knot, row or column is inserted between the last and before-last point, row or column. The lower the tolerance, the closer the insertion to the ends being pulled together, and thus the smaller the region affected. When the tolerance is zero the blended inputs do not change at all: the faces are connected by a straight line and the surfaces by a flatish, linear patch.

Tolerance also affects the size and roundness of the fillet built between non-blended primitives. A zero tolerance yields a short, flat fillet; a unit tolerance generates pointed, non self-intersected fillet.

BIAS*/bias*

Affects only blended primitives by varying the position of the common point, row or column linearly between the two original ends. If the bias is zero, the common part will coincide with the end of the second primitive and the end of the first primitive will be stretched all the way to it. If the bias is one, the common part will coincide with the end of the first primitive and the second primitive will be stretched.

The bias is irrelevant when the blend tolerance is 0.

MULTIPLICITY

Affects the number of knots inserted at the blend point and thus allows for smooth or pointed connections. The connection will be pointed when the multiplicity is *on*. When “blend” is not *on*, an active multiplicity influences the shape and the tightness of the fillet by forcing a multiple knot insertion when *on*.

The fillet tends to be better behaved when multiplicity is *on*. However, this means that the resulting face or surface is built with a discontinuity at the connection points and might not lend itself to point modeling in that area too well.

Multiplicity has no effect on polygons and meshes.

CONNECT CLOSEST ENDS

The Join OP connects the tail of the first primitive with the head on the next primitive, and so on unless this toggle is *on*, in which case the closest ends are chosen instead. For surfaces, this option enables the proximity test in U or V, as specified in the “Direction” parameter below.

DIRECTION

This menu determines the parametric direction of the joining operation, which can be in U or V, and is meaningful only when the inputs are surfaces. The U direction is associated with columns; the V direction refers to rows. For example, joining two surfaces in U will generate a surface with more columns than either input. The number of rows might be higher too, but only if the two inputs have a different number of rows or a different V basis.

JOIN

Can optionally join subgroups of *n* primitives or every *n*th primitive in a cyclical manner. Example: Assume there are six primitives numbered for 0 - 5, and $N = 2$. Then:

- a) Groups will generate 0-1 2-3 4-5
- b) Skipping will generate 0-2-6 and 1-3-5.

N

Determines the number of primitives to be either grouped or skipped. $N \geq 2$.

WRAP LAST TO FIRST

If enabled, it connects the beginning of the first primitive to the end of the last primitive, thus forming a single, closed face or hull. If a single, open primitive exists in the input, it will be closed. The Primitive OP provides a more direct way of closing primitives but offers almost no shape controls.

KEEP PRIMITIVES

If this button is not checked, the input primitives will be deleted after being joined. If checked, they will be preserved.

ONLY CONNECTED

Enabling this option causes only curves and polygons which share a point to be joined.

64.3 MOUSE AND KEYS -

-  or  Click close to the ends of the faces/surfaces to join.
-  Complete the join operation (same as Enter).
-   Pops-up the Operation menu.

64.4 DESCRIPTION -

This Operation allows you to select a sequence of faces or surfaces and connect them, forming a single primitive that inherits their attributes. When you select a primitive click on it close to the end that you want connected. A small white ring is displayed at that end. You can change your mind any time by clicking on another end of that primitive. After you have selected two primitives in this way (both must be faces or surfaces) they will be blended into one according to the parameters currently set (see *Operation Parameters* - p. 606). If only one primitive is selected, clicking on either end will cause the primitive to join to itself. This is one way to wrap a primitive in the modeller using all the flexibility of a join operation. The join is not finalized until you type Enter or click on the right mouse button. Until then you can change both the connection ends as already explained, and the blending parameters.

Once you commit the join, it is automatically selected and a new ring is placed at one end. This allows joining several curves or surfaces in sequence by simply selecting the 'next' curve or surface end to join, completing it (right mouse) and continuing. You can also click on the next pair of faces or curves if you have entered the Join Operation in a "sticky" way (Shift click on its icon); otherwise it returns to the Select Operation.

The pairs joined together in the operation need not be of the same type or degree, nor do they need to have the same number of CVs. For example, it is possible to join a polygon with a quadratic Bézier curve and with a cubic NURBS curve. The Model

Editor converts the simpler types to the more complex - polygon or mesh to Bézier and Bézier and NURBS. In the above example, the new curve will be a cubic NURBS because cubic is the highest of the three degrees and NURBS is the more complex type.

The number of CVs and the type of connection are determined by the “Blend” option, by the tolerance and multiplicity parameters. A blended curve or surface will alter its ends by converting them into a single, common point, row or column respectively. The amount of change can be reduced by lowering the tolerance. When the tolerance is zero the inputs do not change at all: the faces are connected by a straight line and the surfaces by a flatish, linear patch. When not blended, the original shapes are not affected at all, regardless of the tolerance. Instead, the ends are connected by an arc-like fillet. Nonetheless, the result is a single primitive. The multiplicity affects the number of knots inserted at the blend point and thus allows for smooth or pointed connections. When “blend” is not *on*, multiplicity influences the shape and the tightness of the fillet.

64.5 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog. See *Operation Parameters* - p. 606.

BLEND

Specifies the type of join to be applied to the selected pair. If *enabled*, the ends will be brought together to a single point (row or column for surfaces) whose position is interpolated linearly between the original ends. If this button is *disabled*, the original shapes are not affected at all. Instead, the ends are connected by an arc-like fillet.

BIAS

Controls the position of the blended position relative to the ends being connected. If the bias is zero, the first primitive selected remains unchanged and the end of the second primitive stretches to touch the first. If the bias is zero, the end of the first primitive stretches to touch the end of an unaffected second primitive. The bias is inactive if “blend” is *off*.

64.6 JOIN OPERATION MENU -

TOGGLE BLEND B

Switch between a blend and a fillet mode.

FINISH JOINING Enter

Commit the join and start joining a new pair.

64.7 OPERATION PARAMETERS -

TOLERANCE

If “blend” is *on*, the tolerance indicates how much blending can affect the shapes near the ends being connected. The smaller the tolerance, the closer will a point or knot be inserted to the end-points. When the tolerance is zero the inputs do not change at all: the faces are connected by a straight line and the surfaces by a flatish, linear patch.

If “blend” is not *on*, the tolerance acts as a scaling factor to the two ends of the fillet connecting the end-points.

MULTIPLICITY

Affects the number of knots inserted at the blend point and thus allows for smooth or pointed connections. The connection will be pointed when the multiplicity is *on*. When “blend” is not *on*, an active multiplicity influences the shape and the tightness of the fillet by forcing a multiple knot insertion *on*.

The fillet tends to be better behaved when multiplicity is *on*. However, this means that the resulting curve or surface is built with a discontinuity at the connection points and might not lend itself to point modeling in that area too well.

Multiplicity has no effect on polygons and meshes.

65 JOINT OP

65.1 DESCRIPTION

The Joint OP will aid in the creation of circle-based skeletons by creating a series of circles between each pair of input circles. This OP requires at least a pair of circles in order to work correctly.

65.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field causes the OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

DIVISIONS */divs*

Allows you to specify the number of circles between each pair of input circles.

PRESERVE FIRST INPUT

Preserves the first input circle being fed into the OP.

PRESERVE LAST INPUT

Preserves the last input circle.

ORIENT CIRCLES

This helps to create a joint that blends between the input circles without flattening or curving outwards. In order to do this, there may be a reversal of the normal of each input circle. For example, if the normals of the two input circles are facing away from each other, the joint created (if this option was not enabled) would appear to connect the circles the long way around. This option would reorient the joint such that the shorter distance is used to create the joint.

SMOOTH PATH

If not on, the joint circles are blended linearly. Otherwise, they are placed along a cubic piece-wise Bézier curve between the circle centres. This is useful when the input contains more than two circles and the joints must be continuous to each other. If this option isn't enabled, the interpolation may be sharper than desired.

SMOOTH TWIST

Each joint circle is rotated slightly such that its X and Y axis align as it approaches an input circle. This toggle causes the adjustments to be an incremental, or piecewise, Bézier function. Again this is useful for multi-circle inputs.

ALIGN MAJOR AXES

If enabled, this option aligns the first circle's largest axis to the last circle's largest axis. If disabled, the first and last circles' x axes are aligned. This option can help minimize the twist in the joint ellipses between bones.

MINIMUM TWIST

If on, the rotations of the added circles are calculated such that they never rotate further than one half turn in either direction. This leads to a visually continuous layout suitable for creating a skeleton, but will cause problems if the circles are later skinned since the beginnings of each circle may no longer be continuously aligned.

LEFT / RIGHT SCALE

These parameters control the shape of the smooth path, varying the shape of the implied curve from the left or right. If the *Orient Circles* option is on, the sign of the scale has no effect. For a discussion of the relative terms right and left, see *Align OP* p. 430.

LEFT / RIGHT OFFSET

These parameters allow you to override the distance between circles, thereby affecting the shape of the joint.

66 LATTICE OP

66.1 DESCRIPTION

Lattice creates animated deformations of its Data Source by manipulating simpler geometry that encloses the Data Source's geometry.

<i>Data Source</i>	The geometry you wish to deform.
<i>Initial Source</i>	The rest geometry used to determine the influence of points in the deformed geometry.
<i>Deformed Source</i>	A deformed version of the rest geometry. The difference between this and the rest geometry is used to calculate the deformation to apply to the data source.

In Lattice mode, both the initial source and deformed source must be lattices with the point orders in a specified order. This is the order generated by the Box operation when *Use Divisions* is turned on. The number of divisions must match that of the geometry or an error is returned.

In Point mode, both initial source and deformed source must have the same point counts. Note that any points not within the specified radius of a point in the initial source will not be deformed. The amount of deformation applied to a point is based upon the weighted average of the deltas of the initial source to the deformed source. The weighting function is taken from the specified metaball kernel, so behaviour is similar to applying a magnet metaball to every point in the initial source.

66.2 PARAMETERS – LATTICE

GROUP

Subset of points in the first input to be deformed. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

DIVISIONS

/divsx /divsy /divsz

Must be set to match the number of divisions in the lattice grid.

66.3 PARAMETERS – POINTS

REST GROUP

Defines the points in the rest geometry input to use for deformations. Corresponding points in the deformed geometry input are used as the implicit deform group.

KERNAL FUNCTION / POINTS

Which meta kernel to use to determine the influence of a point. Deformation by specifying a Kernal Function and Points makes it easier to deform arbitrary clouds of points, as this makes the topology of the lattice behave more like a metaball rather than as a fixed lattice.

RADIUS */radius*

The size of the points capture regions.

66.4 INPUTS / GEOMETRY TYPES

All input types. Lattices should consist of open polygons, usually created with the Box OP.

66.5 SEE ALSO

- *Box OP* p. 461
- *Bulge OP* p. 467
- *Magnet OP* p. 632
- *Metaball OP* p. 638
- *Spring OP* p. 769

67 LAYER OP

67.1 DESCRIPTION

The Layer operation is designed to allow you to seamlessly work with multiple layers of attributes. This allows you to work with multiple texture coordinate systems, or even multiple coloration layers. If the display option for Multi-pass Texturing is turned on (as it is by default) all of the layers will be drawn on top of one another from the bottom (layer 1) to the top.

Each of the default attributes (i.e. "Cd", "uv", etc) gets renamed according to the current layer. For layer 1 they have their base names. For every other layer, the current layer number is appended. Thus, on layer 5 the colour of the object is determined by "Cd5", not by "Cd" as it usually is. All operations will respect these new names when reading or creating attributes. The Point operation, for example, will use the "Cd5" to determine what \$CR is instead of "Cd".

In any operation with local variables, the current layer is \$CURLAYER, the number of layers is \$MAXLAYER.

67.2 PARAMETERS

CURRENT LAYER */layer*

The current working layer. All further operations in this network will affect the specified layer. If multipass texturing is off, this is what is displayed in the viewport.

MAXIMUM LAYERS */maxlayer*

The maximum number of layers which will be displayed for this operation for multipass texturing.

OVERRIDE MERGE MODE

If set, the default merge mode (on top) will be replaced by a custom defined one for this layer.

SOURCE BLEND

The multiplier factor for the source pixels. Check your OpenGL reference manual for details.

DESTINATION BLEND

The same, but for destination pixels.

CHANNEL MASK

Which channels should be written to by this layer.

CUSTOM MASK */custommask*

Allows any of the 16 possible masks. It's a bitfield with 1 being red, 2 green, 4 blue, and 8 alpha. Add up the numbers of channels you want to write to to get the magic number.

PRE-CLEAR WINDOW

If enabled, the window is cleared (using your mask) before any drawing is done. This is usually used to set up an alpha plane for stenciling.

68 LINE OP

68.1 DESCRIPTION

The Line operation allows you to create lines using polygons, beziers, or NURBs. Lines are defined by a point and a direction (like a vector).

68.2 PARAMETERS

PRIMITIVE TYPE

The type of Geometry to create.

ORIGIN */orig[xyz]*

The start of the line.

DIRECTION */dir[xyz]*

The direction of the line.

DISTANCE */dist*

The length of the line.

POINTS */points*

How many points to make the line with.

ORDER */order*

The order of the NURBs or Bezier curve.

69 LOD (LEVEL OF DETAIL) OP

69.1 DESCRIPTION

The L.O.D. operation is unusual in so far as it does not actually alter any geometry. Instead it builds a level of detail cache for the input object. The cache to be drawn is based upon the distance to the camera. Thus, a complicated object will be drawn with a lower level of detail when it is farther away.

The second input is the rest geometry. If provided, this is the geometry which will be used to do the (expensive) polygon reduction, and only the points of the left input will be used.

69.2 PARAMETERS

STEP % */steppercent*

Each successive levels of detail will contain approximately this percentage of the number of polygons in the higher level of detail.

DIST. THRESHHOLD */distance*

This is the distance from the camera at which full detail will be present.

MINIMUM % */minpercent*

- The objects won't be drawn with fewer than this number of polygons.

STIFFEN BORDER */borderweight*

The amount of weight to avoid erosion of boundary polygons.

EQUALIZE EDGES */lengthweight*

The amount of weight to favour even sized polygons.

PRE-TRIANGULATE */triangulate*

Polygons can only be reduced if they are triangles. This option thus first converts them.

OPTIMIZE RENDERING */tstrips*

When set, triangle strips will be generated and used for drawing.

ONLY AFFECT POLYGON

If this is enabled, only the polygonal portion of the model will be displayed at lower levels of details. Otherwise, all types of surfaces are affected by the distance to the camera./polysonly

69.3 SEE ALSO

- *Visibility OP* p. 858

70 LIMB OP

70.1 DESCRIPTION



The Limb OP creates all necessary geometry for a leg-like limb. It is controlled through inverse kinematics linked to an end-effector (footprint), and create output compatible with the Skeleton OP.

70.2 PARAMETERS – LIMB PAGE

MODEL AXIS

Positions the model along the +X / -X or +Y / -Y axis.

RADIUS */rad1 /rad2*

Controls the size of the circle radii.

KNEE TWIST */twist*

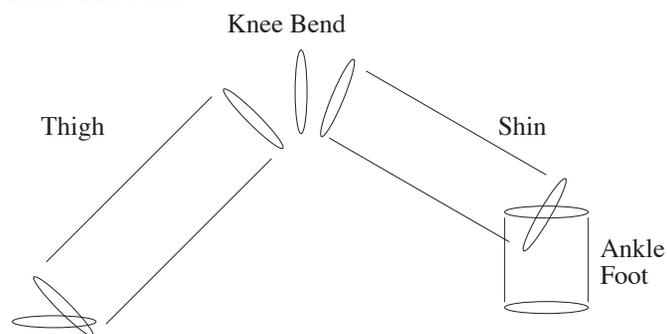
Specifies the rotation angle of the knee joint.

FLIP KNEE

Use this parameter to turn a forward knee into a backward elbow.

70.3 PARAMETERS – LENGTHS PAGE

LIMB LENGTHS



The following parameters control bone lengths, as illustrated above.

- Thigh */thigh*
- Shin */shin*
- Ankle-Foot */foot*
- Knee-Bend */kneedist*

70.4 PARAMETERS – END AFFECTOR PAGE

AFFECTOR OBJECT

Allows the end affector (footprint) to be another object or to be defined by a default box controlled by the transformations below.

TRANSLATE / ROTATE / SCALE /tx /ty /tz /rx /ry /rz /sx /sy /sz

These values apply a transformation on the end affector (footprint) to reorient and reposition it. For a full explanation of transforms, see the *Transform OP* - p. 805.

70.5 USES / WORKS IN RELATION WITH

The limb can be used as input geometry to the Skeleton OP, or as the actual limb geometry. To create useable geometry, append a Convert OP and a Skin OP to the limb.

70.6 TIP

If you are used to working with polygon circles in *action*, and still desire this type of functionality, you can convert them to primitive circles (necessary for the Skeleton, Arm, and Limb OPs) using the Convert OP.

70.7 SEE ALSO

- *Arm OP* p. 433
- *Convert OP* p. 512
- *Skeleton OP* p. 745
- *Skin OP* p. 749

71 L-SYSTEM OP

71.1 DESCRIPTION



L-systems (*Lindenmayer-systems*, named after Aristid Lindenmayer (1925-1989)) allow definition of complex shapes through the use of iteration. They use a mathematical language in which an initial string of characters is evaluated repeatedly, and the results are used to generate geometry. The result of each evaluation becomes the basis for the next iteration of geometry, giving the illusion of growth.

You begin building an L-system by defining a sequence of rules which are evaluated to produce a new string of characters. Each character of the new string represents one command which affects an imaginary stylus, or “turtle”. Repeating this process will grow your geometry.

You can use L-systems to create things such as:

- Create organic objects such as trees, plants, flowers over time.
- Create animated branching objects such as lightning and snowflakes.

THE ALGORITHMIC BEAUTY OF PLANTS

The descriptions located here should be enough to get you started in writing your own L-system rules, however, if you have any serious interests in creating L-systems, you should obtain the book:

The Algorithmic Beauty of Plants,
Przemyslaw Prusinkiewicz & Aristid Lindenmayer,
Springer-Verlag, New York, Phone: 212.460.1500
ISBN: 0-387-94676-4, 1996.

which is the definitive reference on the subject. It contains a multitude of L-systems examples complete with descriptions of the ideas and theories behind modelling realistic plant growth.

71.2 PARAMETERS – GEOMETRY PAGE

TYPE

Provides two options for output geometry:

Tip: Instead of *Tube*, you could also use *Skeleton* with a Polywire SOP.

Skeleton

Creates wire frame geometry. This option is ideal for geometry that is stiff and jagged like lightning or snowflakes. It is also useful to reduce OP cooking time.

Tube

Creates tube geometry. This option can be used with solid geometry that would need smooth curves, like trees or shrubs. Parameters on the *Tube* page are only enabled when this Type is selected.

GENERATIONS */generations*

Determines the number of times to apply the rules to the initial string. This value controls the growth of the L-system. Place a time-based function here to animate the L-system growth.

RANDOM SCALE */randscale*

Random scale as a percentage. This will apply a random scale to the changing geometry's lengths, angles and thickness.

RANDOM SEED */randseed*

Random seed for the OP. This value can be used to select different sequences of random values.

CONTINUOUS ANGLES

Calculates the incremental angles of branches, if a non-integer generational value is used. If the *Generations* field is animating, this should be set to ensure smooth growth.

CONTINUOUS LENGTH, CONTINUOUS WIDTH

Calculates the incremental lengths of the geometry points if a non-integer generational value is used. As with *Continuous Angles*, if the *Generations* field is animating, this should be set to ensure smooth, continuous growth. The *Continuous Width* field applies to tube thickness.

APPLY COLOR

Use a picture or ramp file to apply color to the L-system as it grows.

IMAGE FILE

Defines picture or ramp file to use when the *Apply Color* button is selected. Also see the ' and # turtle operators.

UV INCREMENT */incu /incv*

Defines the default color U, V index increments when the turtle symbols ' or # are used.

POINT WIDTH ATTRIBUTE

This option is useful when outputting to RenderMan and vMantra.

The effect of this parameter is similar to *Tube* output, but instead of tube geometry, it outputs skeletons with a corresponding width at each point. This significantly reduces the amount of geometry output by using RenderMan *RiCurve* primitives with varying width.

rendering l-system curves with width with renderman

Once your L-system geometry is setup with the *Point Width Attribute*, you must append an Attribute OP to it, and in the *RenderMan* Page, set the first line to:

```
Houdini: width
RiName : width
RiType : vertex float (default)
Offset : 0 (default)
```

Then, in the Textport type:

```
setenv RMAN_CURVE_BASIS = cubic
```

(the default can be restored with: `setenv RMAN_CURVE_BASIS = linear`)

Be sure to turn off *Smooth Shading* in the L-system Object (*Shading* page), so it will not add normals and everything will be oriented to the camera as expected; otherwise you get something that looks like ribbons instead of tubes.

71.3 PARAMETERS – TUBE PAGE

The parameters on this page are active only if Geometry page > Type has been set to the Tube type.

Note: If you are outputting to RenderMan, consider using the *Point Width Attribute* (on the *Geometry* page) to output *RiCurve* primitives directly instead of larger amounts of tube geometry.

ROWS / COLUMNS */rows /cols*

The first option sets the number of tube sides and the second sets the number of divisions per step length if tube geometry is selected.

TENSION */tension*

Tension defines the smoothness of branching corners. This parameter applies to tube geometry only.

BRANCH BLEND

Enabling this option allows a child branch to be continuously joined to its parent branch.

THICKNESS */thickinit /thickscale*

The first number defines the default tube thickness; the second number is the scale factor used with the *!* or *?* operator. This option applies to tube geometry only.

APPLY TUBE TEXTURE COORDINATES

When enabled, UV texture coordinates are applied to the tube segments, such that the texture wraps smoothly and continuously over branches.

VERTICAL INCREMENT */vertinc*

Defines the vertical spacing of texture coordinates over tube geometry when tube texture is applied.

71.4 PARAMETERS – VALUES PAGE

STEP SIZE / SCALE */stepinit /stepscale*

The first option allows you to define the default length of the edges when new geometry is generated.

The second option defines the scale by which the geometry will be modified by the “ or _ (double quote, or underscore) turtle operators.

ANGLE */angleinit /anglescale*

The first option defines the default turning angle for *turns*, *rolls* and *pitches*. The second number allows you to enter the scaling factor to be employed when the ; or @ operators are used.

VARIABLE B C D */varb /varc /vard*

Substitutes user-defined b, c and d variables in rules or premise. These variables are expanded and so may include system variables such as \$F and \$T.

GRAVITY

This parameter determines the amount of gravity applied to the geometry via the *T* (tropism vector) turtle operator. Tropism is when a plant bends or curves in response to an external stimulus. L-systems employ a *tropism vector* to simulate this behaviour. The bending is characterised by the fact that the thicker or shorter parts bend less than the longer or thinner parts.

PIC IMAGE FILE

This file specifies a four-channel bitmap image to use with the *pic()* function.

GROUP PREFIX

If the production *g(n)* is encountered, all subsequent geometry is included in a primitive group prefixed with this label and ending with the ASCII value of *n*.

Note: See *Creating Groups within L-systems* p. 629 for how to use L-systems Groups

CHANNEL PREFIX

If the expression *chan(n)* is encountered, it is replaced with the local channel prefixed with this label and ending with the ASCII value of *n*.

71.5 PARAMETERS – FUNCS PAGE

The parameters on this page allow you to stamp your leaf geometry (each copy can be different) as opposed to simply copying them. See the example in *Example – Stamping L-system Leaves* p. 630.

PIC IMAGE FILE

This is the name of the file which the *pic()* function uses.
See *L-system Specific Expression Functions* p. 628.

GROUP PREFIX

Enter a prefix here for the name of groups created using the *g* operator.
See *Creating Groups within L-systems* p. 629 for an example.

CHANNEL PREFIX

Enter a prefix here for the names of channels to reference.

LEAF PARAM A B C

You can determine which parameters are used by leaves.
See *Creating Groups within L-systems* p. 629 for an example.

71.6 PARAMETERS – RULES PAGE

CONTEXT IGNORE

This field contains all characters which are to be skipped when testing context sensitivity in the *Rules* section.

PREMISE

Initial string of characters to which the substitution rules are applied.

RULES 1 - 3

This is where the turtle substitution Rules 1-7, 8-16, and 17-25 are entered.

71.7 RULE SUBSTITUTION

You create the highly structured organic and branching objects using L-systems grammar. An L-system is a process in which a sequence of rules are applied to an initial string of characters to create a new string. To build the geometry, each character of the final string represents one command which affects an imaginary stylus, or “turtle”.

The process begins by examining the first character of the premise string. All sixteen rules are searched sequentially until an applicable rule is found. The current character is then replaced with one or more characters as defined by the rule. The remaining characters in the premise string are replaced in a similar fashion. The entire process is repeated once for each generation.

LIMITATIONS TO RULES

- Polygon {} and branch [] operators can be nested 30 levels deep
- Rules can be 256 characters in length
- Variables can have up to 5 parameters
- Up to 25 rules can be defined

TURTLE OPERATORS

F	Move forward (creating geometry)
H	Move forward half the length (creating geometry)
G	Move forward but don't record a vertex
f	Move forward (no geometry created)
h	Move forward a half length (no geometry created)
J K M	Copy geometry source J, K or M at the turtle's position after rescaling and reorienting the geometry.
T	Apply tropism vector
+	Turn right
-	Turn left (minus sign)
&	Pitch up
^	Pitch down
\	Roll clockwise
/	Roll counter-clockwise
	Turn 180 degrees
*	Roll 180 degrees
~	Pitch / Roll / Turn random amount
”	Multiply current length
!	Multiply current thickness

;	Multiply current angle
_	Divide current length (underscore)
?	Divides current width
@	Divide current angle
'	Increment color index U (single quote)
#	Increment color index V
%	Cut off remainder of branch
\$	Rotate 'up' towards the sun about heading
[Push turtle state (start a branch)
]	Pop turtle state (end a branch)
{	Start a polygon
.	Make a polygon vertex
}	End a polygon
g	Create a new primitive group to which subsequent geometry is added

71.8 PRODUCTION RULE SYNTAX

A Houdini L-system rule is specified as:

$[lc<] pred [>rc] [:cond]=succ [:prob]$

where:

<i>lc</i>	Optional left context
<i>pred</i>	predecessor symbol to be replaced
<i>rc</i>	Optional right context
<i>cond</i>	Condition expression (optional)
<i>succ</i>	Replacement string
<i>prob</i>	Probability of rule execution (optional)

CONTEXT SENSITIVITY

The most basic type of rule is:

$pred=succ$

In this case, a character is replaced with the characters of *succ* if, and only if, it matches *pred*. For example:

Premise: ABC
 Rule 1: B=DOG
 will result in ADOGC

pred can only specify one letter, but left and right context symbols can be specified. The general syntax is $[lc<] \textit{pred} [>rc] = \textit{succ}$. For example:

Premise: ABC
Rule 1: A<B=DOG

again results in ADOGC because B is preceded by A. If the rule were: Z<B=DOG or B>A=DOG the rule would not be applied.

71.9 PARAMETER SYMBOLS

Each symbol can have up to five user-defined variables associated with it which can be referenced or assigned in expressions. Variables in the predecessor are instanced while variables in the successor are assigned.

For example, the rule:

$$A(i, j) = A(i+1, j-1)$$

will replace each A with a new A in which the first parameter has been incremented and the second parameter decremented.

Note that the variables in the predecessor can also be referenced by the condition or probability portions of the rule. For example, the rule:

$$A(i):i < 5 = A(i+1) A(i+1)$$

will double each A a maximum of five times (assuming a premise of A(0)).

Parameters assigned to geometric symbols (e.g. F, +, or !) are interpreted geometrically. For example, the rule:

$$F(i, j) = F(0.5*i, 2*j)$$

will again replace each F with a new F containing modified parameters. In addition to this, the new F will now be drawn at half the length and twice the width.

71.10 OPERATOR OVERRIDES

Normally turtle symbols use the current length/angle/thickness etc. to determine their effect. By providing a turtle operator with an explicit parameter, it will override the value normally used by the turtle operator.

Override parameters for F, f, G, h, H take the form of:

$$F(i, j, k, l, m)$$

i	Override Length.
j	Override Thickness.
k	Override # Tube Segments.
l	Override # Tube Rows.
m	User parameter.

The *k* and *l* override parameters allow dynamic resolution of tube segments.

EXAMPLES

F	Moves forward current length creating geometry
H	Moves forward half current length creating geometry
F(<i>i</i> , <i>j</i>)	Moves forward a distance of <i>i</i> , creating geometry of thickness <i>j</i> .
H(<i>i</i> , <i>j</i>)	Move forward half the distance of <i>i</i> , creating geometry of thickness half of <i>j</i> .
+	Turn by current angle amount.
~	Rotate by random angle.
+(<i>i</i>)	Turn by <i>i</i> degrees.
~(<i>i</i>)	Override random angle with value of <i>i</i> .
\$(<i>x0</i> , <i>y0</i> , <i>z0</i>)	Points the turtle to location (<i>x0</i> , <i>y0</i> , <i>z0</i>)

Given the above, the premise:

F(1) +(90) F(1) +(90) F(1) +(90) F(1)

generates a unit box regardless of the default *Step Size* or *Angle* settings.

LIST OF OPERATOR OVERRIDES

The following list describes the geometric interpretation of parameters assigned to certain turtle symbols:

i overrides current step length, and j the current tube thickness	i overrides current angle
F(<i>i</i> , <i>j</i>)	+(<i>i</i>)
f(<i>i</i> , <i>j</i>)	-(<i>i</i>)
H(<i>i</i> , <i>j</i>)	/(<i>i</i>)
h(<i>i</i> , <i>j</i>)	\(<i>i</i>)
G(<i>i</i> , <i>j</i>)	&(<i>i</i>)
	^(<i>i</i>)
	@(<i>i</i>)
i overrides thickness	i overrides gravity
?(<i>i</i>)	T(<i>i</i>)
i overrides current length	i overrides scale
_(<i>i</i>)	"(<i>i</i>)
i overrides random angle (default 180±[]°)	!(<i>i</i>)
~(<i>i</i>)	;(i)
i overrides scale of input geometry	i overrides group suffix
J(<i>i</i>)	g(<i>i</i>)
K(<i>i</i>)	
M(<i>i</i>)	
i overrides UV increment (absolute value)	
'(<i>i</i>)	
#(<i>i</i>)	

71.11 EDGE REWRITING

In *The Algorithmic Beauty of Plants*, many examples use a technique called Edge rewriting which involve left and right subscripts. A typical example might look like:

```
Generations=10 Angle=90
Premise F(l)
Rules:
  F(l) = F(l)+F(r)+
  F(r)=-F(l)-F(r)
```

However, Houdini doesn't know what $F(l)$ and $F(r)$ are. In this case, we can modify the rules to use parameter passing. For the F turtle symbol, the first four parameters are *length*, *width*, *tubesides*, and *tubesegs*, leaving the last parameter user-definable. We can define this last parameter such that 0 is *left*, and 1 is *right*:

```
Generations: 10
Angle: 90
Premise: F(1,1,3,3,0)
Rules:
  F(i,j,k,l,m) :m=0 = F(i,j,k,l,0)+F(i,j,k,l,1)+
  F(i,j,k,l,m) :m=1 =-F(i,j,k,l,0)-F(i,j,k,l,1)
```

After two generations this produces: Fl+Fr+-Fl-Fr

There should not be any difference between this final string and: F+F+-F-F

Another approach is to use two new variables, and use a conditional statement on the final step to convert them to F:

```
Variable b: ch("generations")
Premise: 1
Rules:
  l:t<b=1+r+
  r:t<b=-1-r
  l=F
  r=F
```

Output:

```
Gen 0: 1
Gen 1: F
Gen 2: F+F+
Gen 3: F+F++-F-F+
```

71.12 EXPRESSIONS

In the earlier example, the expressions $0.5*i$ and $2*j$ are used. In fact, expressions can be used anywhere a numeric field is expected. Currently the following symbols can be used within expressions:

()	brackets for nesting priority
$\wedge + - * / \%$	arithmetic operators
min() max() sin() cos() asin() acos() pic() in()	supported functions
== != = < <= > >=	logical operators
& !	logical operators: and, or, not
b c d	OP b, c, d parameters after expansion
x y z	current turtle position
g	age of symbol
t	time (generations) of L-system
a	OP angle parameter
T	OP tropism (gravity) parameter

The pre-defined variables above should not be used in the arguments of the predecessor. For example:

$A(a,b) = B(a*2,b*2)$ **WRONG** (*a* is the OP Angle parameter)
 $a < b (A,B) = b(A+1,B)$ **RIGHT**

The last statement is correct because *a* and *b* are used as symbols and not variable names. *A* and *B* are correct because variable names are case sensitive.

L-SYSTEM SPECIFIC EXPRESSION FUNCTIONS

$pic(u, v, c)$ Using the image specified with *Pic Image File*, this function returns a normalized value (between 0 and 1) of the pixel at the normalized coordinates (u,v); c selects one of four channels to examine:

- 0 - grey
- 1 - red
- 2 - green
- 3 - blue

$in(q, r, s)$ Given a *MetaTest* input source containing a metaball geometry, this function returns a 1 if the point (q, r, s) is contained within the metaball, and 0 if not. Use $in(x, y, z)$ (the letters x, y, and z are special and contain the X, Y, and Z location of the turtle) to test whether or not the turtle is currently inside the metaball to create pruned outputs.

CONDITIONS

Each rule may have an optional condition specified. The syntax is:

$[lc<] pred [>rc]:cond = succ$

For example, the rule $A:y>2=J$ includes source J at all A 's above the height of 2.

PROBABILITY

Each rule can specify the probability that it is used (provided it is otherwise applicable). The syntax is: $[lc<] pred [>rc]:cond=succ=prob$

For example:

Rule 1: $A=B:0.5$

Rule 2: $A=C:0.5$

will replace A with either B or C with equal probability.

71.13 CREATING GROUPS WITHIN L-SYSTEMS

There is a group operator 'g' which lumps all geometry currently being built into group g . For example:

$g[F]$

lumps geometry from F into a group called $lsys0$. You can set the $lsys$ prefix from the *Funcs* page.

OPTIONAL GROUP PARAMETERS

g takes an optional parameter as well. For example:

$g(1)[F]$

lumps geometry from f into a group called $lsys1$. If no parameter is given, the default index is bumped up appropriately.

The current group container is pushed/popped with the turtle state so you can do things like:

$gF [gFF] F$

the first and last F 's are put into group 0, and the middle FF 's are put into group 1.

EXAMPLE

Note the example:

$gF [FF] F$

The geometry from all four F 's are put into group 0, (pushing the turtle adopts the parent's group).

To exclude the middle FF from the parent's group type:

$gF [g(-1)FF] F$

71.14 CONTROLLING THE LENGTH OVER TIME

To create an L-system which goes forward X percent less for each iteration, you need to start your Premise with a value, and then within a rule, multiply that value by the percentage you want to remain:

```
Premise: A(1)
Rule: A(i)= F(i)A(i*0.5)
```

This way “i” is scaled before A is again evaluated. The important part is the Premise. You need to start with a value to be able to scale that value.

71.15 EXAMPLE – STAMPING L-SYSTEM LEAVES

1. Place a Circle OP, and set the *Number of Divisions* to:

```
param("lsys", 3)
```

It then displays a triangle (3 is default value).

2. Pipe this into the J input of a L-system OP.

If the L-system premise is:

L-system Premise:	Result:
JA	Triangle
J(,4)A	Square
J(,5)A	Pentagram, etc.

This way, you can customize each leaf before it gets copied.

3. Change the Premise and Rule to:

```
Premise: A(0)
Rule1: A(i)=FJ(, , i+3)A(i+1)
```

This creates a line of increasing-order polygons.

4. Finally, we will want to create 20 leaves, and put them all into a Switch OP. Do this by entering the following expression into the Switch OP’s *Select Input*:

```
param("lsys", 0)
```

5. Then in your L-system, *J(,0)* gives you the first OP, *J(,1)* gives you the second, and so on. This solves the problem of a limited number of leaves using only JKM.

Also note that these examples use only the first stamp parameter, you can use up to three parameters:

e.g. *J(, , 1, 2, 3)*

The first two parameters of J, K, M are used to override length and width, like symbol F.

Note: When stamping, \$CY is the copy number as used by the # of copy parameters, and \$PT is the current point number of the template geometry.

71.16 INPUTS / GEOMETRY TYPES

There are four input geometry sources. The Leaf J, K and M inputs are used by the J, K, and M variables for leaf geometry. These sources are scaled, repositioned and copied at the current position of the turtle.

The fourth input is used for a bounding metaball volume which is tested using the *in()* function.

71.17 SEE ALSO

- *Copy OP* p. 519
- *Switch OP* p. 795

72 MAGNET OP

72.1 DESCRIPTION

This OP allows you to affect deformations of the input geometry with another object using a “magnetic field” of influence, defined by a metaball field. It allows the creation of animated bumps and dents within objects, and other special effects.

It is important to note that the actual deformation comes from the Translates of the Magnet OP, and not from the metaball. The metaball defines the area of effect for the Translates of the Magnet OP. The weight of the metaball determines the effectiveness of the Translates within the magnet OP.

The power of the magnet is greatest at the centre of a metaball field and diminishes to nothing at the edge of the field.

72.2 PARAMETERS

DEFORM GROUP / MAGNET GROUP

Allows you to specify a group of geometry to be deformed, and a group that will act as the magnet respectively. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

72.3 PARAMETERS – DEFORM PAGE

TRANSFORM ORDER

Sets the overall transform order for the transformations. The transform order determines the order in which transformations take place. Depending on the order, you can achieve different results using the exact same values. Choose the appropriate order from the menu.

ROTATE ORDER

Sets the order of the rotations within the overall transform order.

TRANSLATE */tx /ty /tz*

These three fields move the Source geometry in the three axes. The Translates of the metaball only affect the position of the area of influence. The influence itself is provided by an imaginary magnet within the Magnet OP itself, and the *attitude* of this influence is determined by the Translates of the Magnet OP.

Note: If the Translate values of the Magnet OP are all zero, the magnet will have no deforming influence. The weight of the Metaball OP scales the influence of the Magnet OP's Translates.

ROTATE $/rx /ry /rz$

These three fields rotate the Source geometry in the three axes.

SCALE $/sx /sy /sz$

These three fields scale the input geometry in the three axes.

PIVOT $/px /py /pz$

The pivot point for the transformations. Not the same as the pivot point in the pivot channels.

72.4 PARAMETERS – ATTRIBUTES PAGE

affect position

Allow the magnet to affect the position of the input geometry.
This is enabled by default.

affect point color

Allow the magnet to affect the point color of the input geometry.

Tip: To control the contribution of each magnet on the surface's Point Color when the *Affect Point Color* option is enabled, set your point colours to black (0,0,0) before the Magnet OP by using a Point OP. The Translate fields in the Magnet OP will then add per-point RGB colour with values of 2,2,2 approaching white.

The scale and rotate channels of the magnet move you about in 3D colour space.
This is not recommended.

affect point normal

Allow the magnet to affect the point normals of the input geometry.

affect point velocity

Allow the magnet to affect the velocity of the input geometry.

72.5 SEE ALSO

- *Metaball OP* p. 638

73 MATERIAL OP

73.1 DESCRIPTION

The Material OP allows the assignment of materials created in the Material Editor to groups of individual primitives.

Note: The *matte* parameter in Objects will override material attributes assigned to geometry using the Material OP.

73.2 PARAMETERS

GROUP

Specifies which group to add materials to. If left blank, the material will be applied to all primitives in the input source. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

To create new materials in Houdini, use the Material Editor. See the *TOPs & SHOPs* sections.

MATERIAL

Select a material here to apply.

73.3 MOUSE AND KEYS -

 Click on a geo object within the Viewport to assign the current material / SHOP in the Material List to the object on which you click.

 or  Click on a geo object in the Viewport to display the name of the Material assigned to it.

73.4 DESCRIPTION -

This Operation allows you to assign and query materials in your scene. Simply select a material or shop from the sub-icons pop-up menu, and click on the object which you want to assign that material with . This changes the *Material* set in the object's *Shading* page.

To display the name of the material or shop currently assigned to an object without changing it, click with  or .

73.5 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog. This displays a blank dialog. The purpose of this is that you are able to call up the parameters dialog, so when you switch to another Operation, it remains and displays the parameters for each Operation as you switch between them.

SHOP / MATERIAL SWITCH

Switches between applying materials (composed of TOPs), and SHOPS.

LIST OF MATERIALS / SHOPS

Choose from a list of active Materials / SHOPS here.

74 MEASURE OP

74.1 DESCRIPTION

The Measure operation performs measurements on primitives. It allows you to calculate area and perimeter of primitives and put the result into attributes.

The area of polygons is only exact when they are planar. Otherwise it is the area of the triangle fan of the polygon from its barycentre.

Bezier and NURBs curves use the measurements of the polygon which shares their control points. Bezier and NURBs patches use the measurements of the mesh which shares their control points.

Tip: To birth particles based on primitives' areas, use one of the following emission types in your source POP: *Edges (attribute)*, *Prim Center (attribute)*, and *Surfaces (attribute)* – these use the 'Distribution Attribute' from each primitive as the probability distribution. See the Source POP for more info.

74.2 PARAMETERS

GROUP

The primitive group to be measured.

TYPE

The type of measurement to be performed. It is one of area or perimeter.

OVERRIDE NAME

Determines if the default attribute name should be used.
The default for perimeter is "perimeter" and for area "area".

ATTRIBUTE

The name of the attribute to store the generated measurement in, if the default is not to be used.

75 MERGE OP

75.1 DESCRIPTION

This OP lets you Merge geometry from different OPs.

75.2 PARAMETERS

INPUT OP

Simply connect the output of two or more OPs to the input of the Merge OP and the geometry will be combined here. Click the up-arrow to move the OP upward in the list, and the “X” to remove the OP from the list.

75.3 INPUTS / GEOMETRY TYPES

Handles up to 9999 inputs. All geometry types are acceptable.

75.4 LOCAL VARIABLES

none.

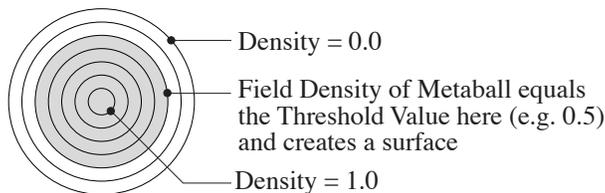
75.5 USES / WORKS IN RELATION WITH

Great for combining geometry from several different OP inputs.

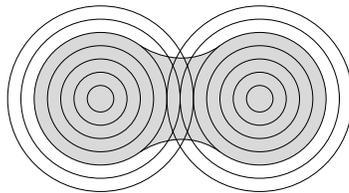
76 METABALL OP

76.1 DESCRIPTION

This OP is for creating metaballs and meta-superquadric surfaces. Metaballs can be thought of as spherical force fields whose surface is an implicit function defined at any point where the density of the force field equals a certain threshold. Because the density of the force field can be increased by the proximity of other metaball force fields, metaballs have the unique property that they change their shape to adapt and fuse with surrounding metaballs. This makes them very effective for modeling organic surfaces. For example, below we have a metaball. The surface of the metaball exists whenever the density of the metaball's field reaches a certain threshold:



When two or more metaball force fields are combined, as in the illustration below, the resulting density of the force fields is added, and the surface extends to include that area where the force fields intersect and create density values with a value of one. For more information on Metaballs, see the *Geometry Types* section.



LEVEL OF DETAIL FOR METABALL DISPLAY

You can change the level of detail of the metaball and NURBS display by adjusting the Level of Detail parameter in the Viewport \oplus > *Viewport* page > *Level of Detail* option. This is described in: *Interface* > *Level of Detail* p. 128.

You can also set this with the scripting command: `viewdisplay`, or by setting the environment variable HOUDINI_LOD. See the *Scripting* section for information on the `viewdisplay` command and Environment Variables. Example:

```
viewdisplay -1 3 Build.panel.world.persp1
```

BETTER METABALL SHADING TIP

Accurate metaball normals will be computed if the *normal* attribute exists when conversion to polygons is done. Thus, to get improved shading on polygonized metaballs, it's a good idea to add the normal attribute (i.e. use a Facet OP) before converting the metaballs.

76.2 PARAMETERS

RADIUS */radx /rady /radz*

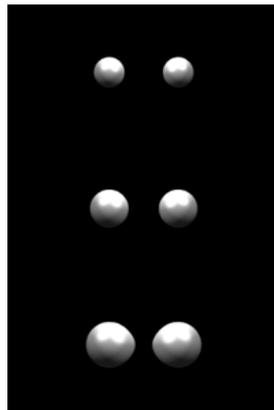
Controls the radius of the metaball field.

CENTER */tx /ty /tz*

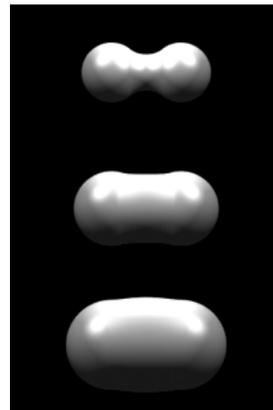
Metaball center in X, Y and Z.

WEIGHT */metaweight*

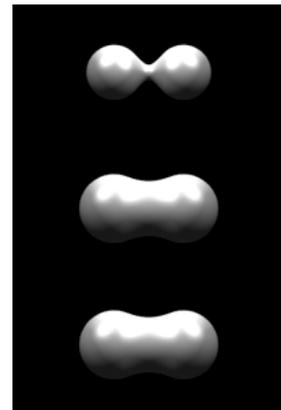
Defines the weight of the Metaball iso-surface within metaball field. An increase in weight makes the density of the metaball greater, and thus the defined implicit surface of it and surrounding metaballs will be enlarged.

KERNEL FUNCTION

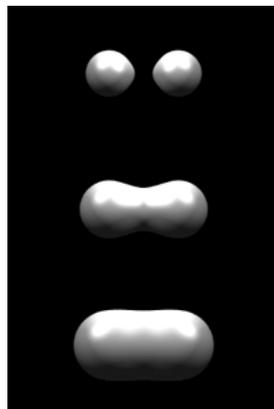
Blinn



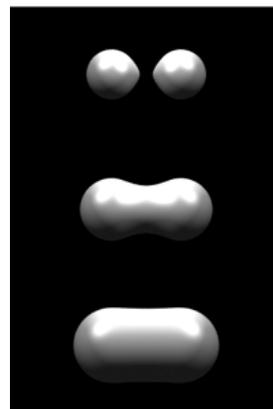
Elendt



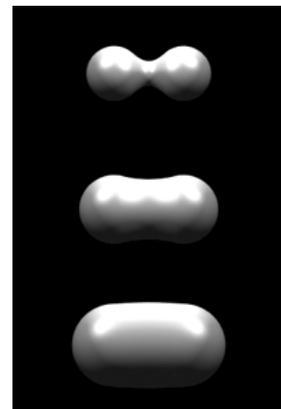
Hart



Links



RenderMan



Wyvill

There are six different metaball interpretations: *Wyvill*, *Elendt*, *Blinn*, *Links*, *RenderMan*, and *Hart*. See the *Geometry Types* section for illustrations of the differences between these.

XY / Z EXPONENT */exp_{xy} /exp_z*

The XY exponent determines inflation / contraction in the X and Y axes. The Z exponent determines inflation / contraction in the Z axis.

What is an Exponent?

In the instance of metaballs, the exponent determines the inflation towards “squarishness” or contraction towards “starishness” as described below:

Value > 1	Results in metaballs that appear more like a “star”.
Value < 1	Results in metaballs that appear more “squarish”.
Value = 1	Results in metaballs that appear spherical.

76.3 USES / WORKS IN RELATION WITH**MODELLING ORGANIC EFFECTS**

This OP is ideal for creating organic models, liquid effects and other metaball surfaces.

FORCE FIELDS AND PRUNING VOLUMES

This OP is also used to define a force field area for the *Force* OP which is used in conjunction with the *Particle* and *Spring* OPs – see also POPs in *Ref > Particles*.

Also, as a MetaTest input for pruning L-systems. This is typically done by copying metaballs to every point of the deforming a polygonal/mesh input source with a Copy OP. When doing this, it is a good idea to pass any deforming geometry through a Facet OP and *Consolidate Points* to minimise the number of points you need to copy to. If converting from spline surfaces, you may want to refine your surfaces first to get a smooth distribution of metaballs.

Tip: If you are using metaballs to create a deforming attractor for the Particle OP, you may want to use a Group OP to define a bounding area, and copy the metaballs only to the points included in that group. This makes it much easier to control and animate the attractor area.

76.4 SEE ALSO

- *Force OP* p. 584
- *L-system OP* p. 618
- *Magnet OP* p. 632
- *Particle OP* p. 653
- *Spring OP* p. 769

76.5 MOUSE AND KEYS -

none

Volatile key

Ctrl 

Pops-up the Operation menu.

76.6 DESCRIPTION -

This Operation is used to create metaballs and metasurfaces. If you click and drag the mouse on the Construction Plane, it generates a metaball whose X radius is specified by your drag. Metasurfaces are generalized metaballs meaning that they aren't necessarily ellipsoidal.

Each metaball has a “sphere of influence”. When two metaballs extend into each other's sphere of influence, they react in a way similar to drops of water – the surface tension works to form a smooth bridge between them. This is useful for making organic “blobby” shapes which meld into each other. For more information on metaballs, see the *Geometry Types* section.

You can see a metaball's sphere of influence by turning on Display Hulls in View Options dialog (see *Viewport*).

Clicking the mouse button without dragging places a metaball with radii as specified in the Parameters dialog box (default of 1) at the location of the mouse click. The XY radii are aligned with the Construction Plane axes.

Typing **Enter** places a metaball whose size and position are specified in the Parameters dialog. The radii are aligned with the Construction Plane axes.

If an odd aspect ratio was previously entered in the Parameters dialog, clicking and dragging will produce metaballs which maintain that aspect ratio. This can be reset by clicking on the *Reset Radii* button.

Note: To change a metaball's level of detail, use Viewport options > *Viewport* page > *Level of Detail* option. This is described in: *Interface* > *Level of Detail* p. 128.

76.7 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog for this Operation. See *Operation Parameters* - p. 642.

RESET RADII

Changes the radii of the metaballs back to 1.

76.8 META-SURFACE OPERATION MENU

Call up the menu for this Operation by using **Ctrl** .

BUILD DEFAULT METABALL *Enter*

Places a metaball at the place currently specified by the center / radius settings in the Parameters dialog. The X and Y radii of the metaball are aligned with the X and Y axes of the Construction Plane.

76.9 OPERATION PARAMETERS -

RADIUS

This is the X radius of a metaball that is placed if you click on the Construction Plane without dragging. If you click and drag, the size of the metaball is over-ridden by the amount of drag. Entering non-equal values in the XYZ fields results in ellipsoidal shapes.

The X radius is defined by the distance dragged from the centre, while the Y and Z radii vary proportionally with the X / Y and X / Z ratios in the parameter dialog.

CENTER

Determines the location of the center of the metaball. This value is updated whenever you click (and drag) to create a metaball. A new metaball will be positioned here if you type *Enter*.

WEIGHT

Specifies the weight of a metaball. Instead of typing, you can use the slider to enter values quickly. The weight of a metaball determines how much “pull” it has on surrounding metaballs, in order to cause them to merge together (see *Geometry Types* section for details).

KERNEL FUNCTION

This menu provides you with the several options which determine the quality and speed of metaball generation. See *Kernel Function* p. 639 for illustrations, or for a complete discussion – *Metaballs* p. 227 in the *Geometry Types* section.

Note: Custom Metaball Kernel functions can be created with the optional HDK (Houdini Developer Kit) – contact Side Effects Software support for information.

XY EXPONENT

The XY exponent determines inflation / contraction in the X and Y axes.

Z EXPONENT

The Z exponent determines inflation / contraction in the Z axis.

WHAT DOES AN EXPONENT DO?

In the instance of metaballs, the exponent determines the inflation towards “squarishness” or contraction towards “starishness” as described below:

Value > 1	Results in metaballs that appear more “squarish”.
Value < 1	Results in metaballs that appear more like a “star”.
Value = 1	Results in metaballs that appear spherical.

77 MIRROR OP

77.1 DESCRIPTION

The Mirror Operation duplicates and mirrors geometry. If the point normal attribute exists, it is also mirrored for each point. Quadrics and pasted surfaces are also supported.

77.2 PARAMETERS

GROUP

The primitives to mirror

ORIGIN

The origin of the mirror plane

DISTANCE */dist*

Distance of mirror plane along its normal.

DIRECTION

Normal direction of the mirror plane.

REVERSE NORMALS

Reverses the normals of the mirrored geometry. Reverse U and Reverse V options apply only to mesh-type primitives (e.g. such as NURBS surfaces)

KEEP ORIGINAL

Preserves the input geometry. If unchecked, only the selected geometry will remain.

CONSOLIDATE SEAM

Consolidates points along the mirror. New points are only consolidated with their mirrored counterpart.

77.3 SEE ALSO

- *Copy OP* p. 519
- *Duplicate OP* p. 551

78 MODEL OP

78.1 DESCRIPTION

This OP is designed specifically to hold raw model geometry. It holds modeled data and cannot be unlocked – protecting you from losing your model data.

Note: This OP is obsolete, and is only included for backwards compatibility. The full functionality of this OP has replaced by In-Viewport editing – you don't need it any more. It will be removed in the next version of the software – You have been warned.

78.2 SEE ALSO

- *Curve Operation* - p. 527
- *Extrude OP* p. 563
- *Force OP* p. 584
- *Rails OP* p. 713
- *Skin OP* p. 749
- *Sweep OP* p. 791

79 NETWORK OP

79.1 DESCRIPTION

The Network Operation transmits and receives geometry data over a network connection. All points, primitives and attributes are sent (selections are not). This allows geometry to be sent from one Houdini session to a remote Houdini session.

The transmitting Network operation sends the data whenever it is cooked, whether or not the geometry has changed. The receiving Network operation receives data whenever it is cooked (it is also time dependent, so changing the time will check if data is received). Only the most recent full transmission is received; geometry does not stack up at the receiving end.

By default, the transmitting Network operation creates a socket at the port specified, and the receiving Network attempts to connect to that socket. You may reverse this, so the receiving Network creates the socket, by turning off the parameter "Default Client/Server Setup". Both the transmitting and receiving Network operations must use the same client/server setup (both on or both off). Normally you will not need to modify this parameter.

A network address is specified as a regular domain name ("mycomputer", "mycomputer.mycompany.com"). The port number is any port between 5000 and 10000. The port numbers at the transmitting and receiving ends must match.

79.2 PARAMETERS

MODE

Sets the Network communications mode to Transmit or Receive.

DEFAULT CLIENT/SERVER SETUP

If on, a socket is created at the transmitting Network operation (see above).

ADDRESS / PORT

Which remote address to connect to. If receiving and the default client/server setup is used, this must be the host address of the remote machine. If you are not using the default setup, the transmitting end must specify the remote address. If the address is omitted, the local host is assumed.

The *Port* specifies the port to use as the server port, or the port to connect to.

ACTIVE

Data is only transmitted or received when active is on.

When receiving, you will lose the transmitted data if this is disabled.

80 NULL OP

80.1 DESCRIPTION

This operation does nothing. It acts simply as a placeholder to make wiring networks easier, or to allow you to reference a single output SOP without having to update all the references to it if the real output SOP changes.

This SOP references its input, so it does no cooking, nor does it increase memory usage.

80.2 SEE ALSO

- *Object merge OP* p. 648

81 OBJECT MERGE OP

81.1 DESCRIPTION

This OP allows you to merge the geometry of different geometry objects into the SOP network in which it is contained.

81.2 PARAMETERS

NUMBER OF OBJECTS

Controls the number of object merge parameters that will be present.

Clicking the *More* or *Less* buttons provides additional parameters to specify more or less merge objects.

ENABLE MERGE

If set, the object will be merged.

OBJECT

The path to the object or SOP to merge. If a SOP is specified, that specific SOP will be merged. If an Geometry Object is specified, the display SOP will be used for display and the render SOP at render time. For explicit control, the tokens “__display_sop__” and “__render_sop__” can be used. A blank path will mean no merge.

TRANSFORM OBJECT

This allows you to transform each object to a common space. The transform object is the one to which all transforms will be relative to.

If it is blank, no transformation is done and objects are imported in world space. If it is the object which contains this SOP, all merged objects will be properly transformed into this object’s space. This behaviour also can be gained by specifying “.” on the path. If a SOP is specified, the object which owns that SOP will be used.

81.3 SEE ALSO

- POP Merge
- POP Network
- Null SOP

82 PAINT OP

82.1 DESCRIPTION

The basic operation of the Paint brush is to apply a stencil to the incoming geometry. Then, paint is applied to the geometry points through that stencil.

Note: Be sure to see: Interface > Brush Tools p. 139 for useful info!

82.2 PARAMETERS – OPERATION PAGE

OPERATION

The operation used by Apply To All.

MERGE MODE

How the color is to be applied to the surface.

ACCUMULATE TO STENCIL

By default, the operation is applied and the stencil cleared after every brush stroke. With this set, the stencil is not cleared until "Apply and Clear Stencil" is pressed.

APPLY & CLEAR STENCIL

Applies the operation and clears the stencil. This is equivalent to right-clicking in the viewport.

FOREGROUND/BACKGROUND */fg /bg*

Colors to apply unless bitmap stamping is enabled. If "Accumulate To Stencil" is on, the left mouse does foreground and the middle mouse erases.

STAMP BITMAP INSTEAD OF FG/BG COLOR

If set, the color channels of the bitmap are used directly rather than the brush color.

APPLY TO ALL

Paints all of the selected geometry.

RESET ALL CHANGES

Restores geometry to initial state.

CREATE MISSING COLOR AND ALPHA ATTRIBUTES

Adds the "Cd" and "Alpha" point attributes if not present and not overridden.

WRITE ALPHA

Use the brush to write into the point alpha channel.

OVERRIDE COLOR

If enabled, it will paint onto the specified attribute rather than the default "Cd".

OVERRIDE ALPHA

Same, but changes the "Alpha" attribute name.

VISUALIZE ATTRIBUTE

Whether to do a false color visualization of the overridden attribute and what range to map to.

VISUALIZE MODE

How to map the attribute value to a colour.

82.3 PARAMETERS – BRUSH PAGE

SHAPE

The basic shape of the brush: circle, square, or bitmap.

BITMAP

What bitmap to use. The alpha channel becomes the brush.

RADIUS */radius*

The radius of the brush.

BRUSH ANGLE */brushangle*

How far to rotate the brush.

BRUSH SQUASH */squash*

Amount to squash the brush in the y direction before rotation.

OPACITY */opacity*

The amount to affect the stencil mask.

BRUSH SPLATTER */brushsplatter*

A random noise in the brush's opacity based upon the position on the brush.

PAPER GRAIN */papergrain*

A random noise on the object's stencil mask based on the object position.

SOFT EDGE */softedge*

Percentage of the brush to be rolled off.

KERNEL FUNCTION

Which metaball kernel to use for the roll off.

UP VECTOR TYPE

How the brush should be oriented on the surface:

Stroke Direction Oriented in the direction in which the brush moves.

Fixed Oriented as specified in the Up Vector field.

UP VECTOR */upvector*

The fixed up vector to orient brush to.

82.4 PARAMETERS – STROKE PAGE

Note: Be sure to see: Interface > Secrets of the Brush Page p. 142 for useful info!

ORIENT BRUSH TO SURFACE

Switches between the brush being perpendicular to the surface or always oriented along the view direction. If you are having trouble with a shaky brush, try turning this off.

REALTIME MODE

Set this to paint geometry as it's deforming.

The last few parameters on this page are read-only parameters. While it is theoretically possible to manually apply brush strokes by adjusting these parameters, that would be excruciatingly difficult. Their primary use is to provide feedback as to where the brush is currently hitting your geometry. This can be used as a sort of 'geometry eyedropper' when you want to quickly investigate some troublesome geometry.

DIRECTION

This is the normal on the hit primitive at the hit position.

HIT LOCATION

This is the position in space where the brush ray hit the geometry.

HIT PRIMITIVE

This is the primitive number that was hit by the brush ray.

HIT UV

These are the parametric coordinates where the primitive was hit. They are the UV coordinates used to evaluate the surface with a prim call, for example, and is not to be confused with any "uv" texture coordinates that may be present.

EVENT

This is used internally to track the current state of the drawing mode. No-op is the default and is used to prevent any accidental writes using out of date or irrelevant hit information.

82.5 SEE ALSO

- *Capture Paint OP* p. 480
- *Comb OP* p. 504
- *Sculpt OP* p. 735

83 PARTICLE OP

83.1 DESCRIPTION

This OP is used for creating and controlling motion of particles for particle systems simulations. Particle systems are often used to create simulations of natural events such as rain and snow, or effects such as fireworks and sparks. In Houdini, the points of the input geometry are used for the starting positions of the particles. Each point of the input can be affected by external force (gravity) and wind. Particles can collide and bounce off other objects (set with the *Collision Source*). They can also bounce off, or die at, *limit planes* set in X, Y and Z.

Particles have various attributes that regular geometry does not have, such as: *velocity*, *life expectancy* and *age*. These attributes are carried with each point in order to carry out the simulation.

Note: Using the Particle OP is adequate for many purposes, and the *only* way in Houdini Select. But if you need to setup a more complex particle system (more than one collision object and/or attractor), you should use POPs (Particle Operators) instead. Refer to the *POPs* section for details.

83.2 PARAMETERS – OPERATION PAGE

BEHAVIOUR

Particle System Act like a particle system, where particles are born from the input points.

Modify Source Geometry Deforms the source input geometry.

Tip: You can also instance geometry to particles in the Object Editor. Make one object the particle system, and another the geometry to instance. Then select the geometry object in the particle system object's *Render* page under *Particle Geo*. This is by far the most efficient way of animating many copies of geometry with a particle system.

START TIME */timestart*

The start time is the time at which the simulation starts cooking. Before this, the particle system will be empty. The default is *Tstart* (frame one).

PREROLL TIME */timepreroll*

How many seconds of the simulation to bypass, after the reset time is reached. For example, if you put the number 33 into this field (and reset is at *Tstart*), frame one will show the simulation that was at a time of 33 seconds. In other words, the first thirty-two seconds have been bypassed, and the time at thirty-three seconds is shifted to frame one. The first thirty-two seconds must still be calculated in order to compute the status of the points, so you will notice some delay upon reset.

TIME INC */timeinc*

The *Time Inc* parameter determines how often to cook the OP. By default, this parameter is set to *1/\$FPS*. This means that the OP will cook once for every frame. When complex dynamics are involved, the OP may require more frequent cooking for increased mathematical accuracy. To get sub-frame accuracy in the cooking, set the *Time Inc* to something smaller than *1/\$FPS*. For example, setting the *Time Inc* to *0.5/\$FPS* will mean that the OP gets cooked twice for every frame.

■ **Note:** Never set this parameter to be greater than *1/\$FPS*.

JITTER BIRTHS

This allows you to jitter the location pixels of each particle as they are born.

ACCURATE MOVES

This option makes the particles move more accurately between frames by calculating their trajectories for fractional frame values.

REMOVE UNUSED POINTS

Removes all unused points from the input geometry. This is provided as an explicit option instead of happening automatically because it saves the time needed to purge the points from memory during the simulation. This prevents unnecessary slow-downs during the simulation.

ATTRACTOR USE

All Points

All point attractors affect all particles (or points).

Single point per particle

When enabled, each particle is assigned a single attractor point, and is affected by only that point. Assignment is done by point number modulo the total number of attractor points.

83.3 PARAMETERS – FORCES PAGE

EXTERNAL FORCE */externalx /externaly /externalz*

Forces of gravity acting on the particles. When drag is zero, the particles can accelerate with no limit on their speed.

WIND */windx /windy /windz*

Wind forces acting on the particles. Similar to external force. Using wind (and no other forces, such as turbulence), the particles will not exceed the wind velocity.

discussion – wind vs external force

The application of external force directly affects a particles' acceleration, the rate of which is determined by the mass ($F = \text{Mass} \times \text{Acceleration}$). Wind is an additional force, but one that is velocity sensitive. If a particle is already travelling at wind velocity, then it shouldn't receive any extra force from it. This implies a maximum velocity when using Wind on its own.

An increase in mass impedes acceleration for a given constant force. Drag is a force opposing the direction of motion which is velocity sensitive, i.e. the larger the velocity, the greater the effect of drag. Its useful for limiting the velocity of particles.

TURBULENCE */turbx /turby /turbz*

The amplitude of turbulent (chaotic) forces along each axis. Use positive values, if any.

TURB PERIOD */period*

A small period means that the turbulence varies quickly over a small area, while a larger value will cause points close to each other to be affected similarly.

SEED */seed*

Random number seed for the particle simulation.

83.4 PARAMETERS – PARTICLES PAGE

ADD PARTICLE ID

Adds an ID number to each particle as it is born.

ADD MASS ATTRIBUTE

When selected, calculates the mass of the particle, as specified in the mass field.

MASS */mass*

The relative mass of each particle. Heavier particles take longer to start moving, and longer to slow down.

ADD DRAG ATTRIBUTE

When selected, calculates the drag coefficient of the particle, as entered in the drag field.

DRAG */drag*

Drag of each particle.

BIRTH */birth*

The number of particles born each second. Particles are not released in clusters. They are born at random times during the first frame of their existence. Their birth time is set randomly during the first frame.

LIFE EXPECT */life*

How long each particle will exist, in seconds. The default is 1000 seconds, which is roughly sixteen and a half minutes. As most animations are considerably shorter than this, use small values such as one or two to see the effect.

LIFE VARIANCE */lifevar*

Variance of a particle's life expectancy in seconds. If life expectancy is one second, and the variance is zero seconds, each particle will live exactly one second. If variance is set to 0.5, then some particles will live only a half second, while others will live a second and a half. The rest will live some time in-between. This randomness gives a more natural look to the particle births.

83.5 PARAMETERS – LIMITS PAGE

+ / - LIMIT PLANE */limitposx ... /limitposz /limitnegx ... /limitnegz*

The particles will die or bounce off the limit planes when it reaches them. The six limit plane fields define a bounding cube. The default settings are 1000 units away, which is very large. Reduce the values to about one to see the effect.

HIT BEHAVIOUR

Control over what happens when a particle hits either the six collision planes or the collide object. The options are:

Die on Contact The particles will disappear when they collide with the Collision input.

<i>Bounce on Contact</i>	The particles will bounce upon contact with the Collision input.
<i>Stick on Contact</i>	The particles will stick to the Collision input.

GAIN TANGENT / NORMAL */gaintan /gainnorm*

Friction parameters which can be regarded as energy loss upon collision. The first parameter affects the energy loss (gain) perpendicular to the surface. 0 means all energy (velocity) is lost, 1 means no energy is lost perpendicular to surface. The second parameter is the energy gain tangent to the surface.

- 1 and 1 means nothing is lost or gained.
- 0.1 and 1 cause the particles to strike the surface and dribble along it, like rain atop a roof.
- 1 and 0 makes them bounce perpendicular to the surface, no matter what angle they came in at.
- -1 and 1 makes the particle bounce back from whence it came.
- Gains greater than 1 cause energy gain (like a pinball machine bumper).

SPLIT

<i>No Splitting</i>	The default setting, this keeps the particles intact.
<i>Split on Contact</i>	The particles will split upon contact with the Collision Source.
<i>Split on Death</i>	The particles will split when they die.

MIN/MAX SPLITS */splitmin /splitmax*

When a particle splits, it splits into a number of other particles. The number of particles is randomly set between this range.

SPLIT VELOCITY */splitvelx /splitvely /splitvelz*

Each split particle is given this base velocity.

VELOCITY VARIANCE */splitvarx /splitvary /splitvarz*

This is a random amount that is added to the split velocity. When creating fireworks, the variance is large while the velocity is low. When rendering raindrops splashing, the split velocity is large in Y, and the variance in X and Z causes the particles to bounce up – but randomly – in the XZ plane.

83.6 PARAMETERS – RENDERING PAGE

PARTICLE TYPE

Though particles are displayed in the Viewport as streaks with a greater brightness at the head of the particle, this parameter controls how the particle system should be rendered:

<i>Spheres</i>	Each particle is rendered as a sphere.
<i>Disks</i>	Each particle is rendered as an oriented disk.
<i>Lines</i>	Each particle is rendered as a line connecting the start and end points between the particle's position on subsequent frames.
<i>Tubes</i>	Each particle is rendered as an open tube.
<i>Capped Tubes</i>	Each particle is rendered as a capped tube.
<i>Rounded Tubes</i>	Each particle is rendered as a rounded tube.

Tip: If you are outputting to RenderMan (prman 3.7 or greater), set your particle type to *Render as Line* for RiCurve() and to *Render as Disk* for RiPoint() which render very quickly.

PARTICLE SIZE */prsize*

How big the particles are. For spheres, disk and tubes this will be the radius.

PARTICLE BLUR */prblur*

This determines how long the particles will appear when rendered. For end to end connectivity, this should be set to $1/FPs$. However, it is possible to stretch the particles by increasing, or shrink by decreasing this value. The view will update to display these changes.

SPHERE NORMAL

When rendering as disks, fake the surface normal to approximate a sphere.

renderer support

mantra and RenderMan support the following render types:

Primitive Type	mantra	RenderMan
Sphere	•	•
Disks	(as spheres)	•
Lines	(as tubes)	•
Tubes	•	•
Capped Tubes	•	•
Rounded Tubes	•	•

This table is only in regards to particle renderings. When something doesn't match, the particles will still be rendered, but not necessarily what you'd expect.

Note: When rendering disks with *mantra*, the sphere normal is always used.

NOTE ABOUT RENDERED PARTICLE ATTRIBUTES

If the point attribute *pscale* (single float) is found, this will be used to scale each particle independently. This can be used to get different sized particles for rendering. The *pscale* attribute applies a uniform scale to each particle. When rendering sphere particles, the radius of the sphere will be scaled by the value, when rendering tubes, the radius of the tube will be scaled by the value. When rendering using instance geometry, each instance will be uniformly scaled by the value of the attribute. This is implemented for both *mantra3* and RenderMan.

You can create/modify the *pscale* attribute using the Point OP (*Point OP* p. 667).

83.7 INPUTS / GEOMETRY TYPES

PARTICLE SOURCE

Any geometry that provides points, e.g. polygon sphere, mesh.

COLLISION OBJECT

This input defines an object for the particles to collide with. When this happens, the particles can either die, stick or bounce, or spawn new particles. It is important to note that when the collision object is deforming, collision detection may fail, causing some particles to “leak through” the collision object.

FORCE

The Force input accepts input from a Force OP which uses a metaball shape as a force field allowing particles to be sent into vortices or accelerated along an axis. Refer to the *Force OP* p. 584 for further explanation.

Note: The Particle OP will use point normals as initial particle velocity if point normal attributes exist *and* there are no point velocity attributes in the incoming data. If you add velocity attributes to the points, the point normals are ignored.

83.8 SEE ALSO

- *Attribute OP* p. 436
- *Force OP* p. 584
- *Metaball OP* p. 638
- *Point OP* p. 667
- *POP OP* p. 694

84 PARTITION OP

84.1 DESCRIPTION

The Partition SOP places points and primitives into groups based on a user-supplied rule. For example, in order to put each point in its own group named

"pt_<point number>", enter the rule: "pt_\$PT".

The rule can be any valid Houdini expression. Rules should evaluate to valid group names. The result of a rule may be modified to make it a valid group name (for example a result of "1" will be turned into the group "_1").

Standard local variables are available as well as local variables created with the *AttribCreate* SOP.

84.2 PARAMETERS

GROUP

The primitives or points considered for the operation.

ENTITY

The type of the entity specified in the group parameter.

GEOMETRY TYPE

The type of primitives to consider for this operation. Only applicable when Entity is set to Primitives.

RULE

The rule with which to evaluate each primitive or point. The result of the rule is the name of the group in which the primitive or point will be placed.

84.3 LOCAL VARIABLES

The standard local variables are usable in this operation.

84.4 SEE ALSO

- *Group OP* p. 592
- *Attribute Create OP* p. 442

85 PASTE OP

85.1 DESCRIPTION

The Paste OP is a local refinement tool that doesn't increase the complexity of the base surface, yet allows detail to move freely on the base surface.

A feature of the Paste OP is that regardless of the pasting method, the base hierarchy will always keep its primitive numbers. The features will be added to the existing model, so their numbers will always be higher than those of the base(s). This allows you to change the resolution of the animation at the base level and animate base points at various resolutions without worrying about point-number changes.

For a discussion on Pasting, also see *Pasting* p. 247 in the *Geometry Types* section.

TWO WAYS OF BUILDING A PASTE HIERARCHY

There are two ways of building a paste hierarchy; from outside the base surface, and from inside the base surface.

adding detail from outside

This is done using the Paste OP, with two inputs: the feature and the base.

- Parametric Paste
- Along Vector (Projective) Paste

1. In a Parametric Paste, the Paste OP places the feature on an area of the base surface delimited by four user-defined isoparametric curves. The feature is thus aligned with base surface isoparametrically.

The advantage of Parametric Pasting is that the feature is guaranteed to “land” on the base within a well determined parametric area regardless of the base's shape, position and orientation; also, the continuity between feature and base is enhanced by the parametric alignment between the two surfaces.

2. In a Projective Paste, the four corners of the feature surface are projected onto the base surface first. Then the entire feature is moulded onto the base such that its corners match the four projected points. The feature is not aligned with the base isoparametrically.

The advantage of the Projective Paste is that it applies the feature onto the base intuitively along a vector, without any parametric alignment, generally producing a mould that is similar in shape and orientation to the original (unpasted feature).

growing detail from within

If you use only one input to the Paste OP, you can still create a pasted surface using a method called “spawning”. This technique extracts a portion of the base surface and turns it into a pasted surface that shares the base's shape and underlying domain in that area (much like an onion peel).

The new surface can be further refined and modeled to generate the desired detail; it can be spawned recursively to add even more detail. This new surface may be lofted

above the base surface by the amount specified by the *Height* parameter. This is an easy way to build offset surfaces. The advantage of the spawning technique is that it guarantees perfect geometric and texture continuity between feature and base.

SOME NOTES ON GETTING GOOD PASTED SURFACES

- The flatter the base surface, the less distorted the pasted surface will be.
- The more tentacles on the feature surface (i.e. the more refined the feature), the better its moulding on the base surface.
- The more vertical the initial tentacles, the more closely the pasted surface will follow the features of the base.
- The shorter the tentacle, the closer to the base surface it will be. Therefore, in general it is better to start with a flat base surface, and deform it in the middle, leaving the edges so their slope is flat. This yields better boundary continuity across feature and surface.
- It is also good to ensure that the feature starts out as a rectangular grid and all the interior deformations don't "spill-out" of the rectangular area.

Tip: If the feature doesn't have enough resolution (i.e. tentacles), you may have a problem with boundary continuity, in which case you can add a Refine OP to the feature before doing the paste, or increase the number of boundary isoparms with the *Belt* parameters.

SURFACE DELETION

The deletion of surfaces is available in the *Delete OP* p. 543 and *Unpaste OP* p. 831, as well as in the Model Editor.

CURRENT LIMITATIONS

- You cannot paste across multiple paste hierarchies.
- Most, but not all OPs support the "Pasted" primitive type. Those that do not will either ignore the hierarchical primitive or delete it.

85.2 PARAMETERS

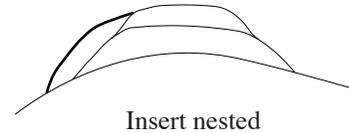
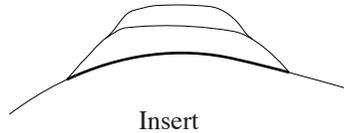
FEATURE / BASE GROUP

You can optionally enter a group name in either of these two fields to specify a Feature and/or a Base to act on. The Base group must contain no more than one spline surface or paste hierarchy. The Feature group can contain any number of spline surfaces or paste hierarchies. If a feature surface is already pasted, it's entire hierarchy will be pasted onto the base.

Note: A paste hierarchy created with the Paste OP displays it's the paste hierarchy's primitive number in brackets (e.g. (5)) when primitive numbers are enabled in the Viewport options. However, the brackets should not be used when specifying the primitive number. It should be treated just like any other primitive number.

OPERATION

- Paste* Applies a feature surface or hierarchy onto the base.
- Insert* Applies a feature surface or hierarchy underneath the current hierarchy of features, but above the root surface (usually the base, unless a different base group is specified).
- Insert nested* Applies a feature surface across the current hierarchy of features.



HEIGHT

Amount of elevation from the base surface; both positive and negative. In the areas where the feature isn't flat, there is an inherent elevation. The height is added to that elevation.

FLIP PASTING DIRECTION

Inverts the pasted feature to create either bumps or dents on the base surface.

TRIM BASE

This parameter removes the area of the base surface lying under the newly pasted feature. Current limitation: if the feature lands on multiple surfaces, Houdini trims out only the surface that the feature lies on *entirely*.

When enabled, the scaling slider beside it can be used to adjust the size of the trimmed area. A value of 1 (default) makes the trimmed-out area exactly as large as the domain covered by the feature.

BELT WIDTH / DIVISIONS

To increase the refinement of the feature around its perimeter (also known as a "belt"), you can specify a number of isoparms to add in U and V, and a parametric width around the boundary in which to insert these isoparms. If the width is zero, no isoparms will be added. This yields better boundary continuity with the base.

Belt refinement is done in such a way so as to ensure that the divisions are applied evenly in the domain of the surface and not too close to existing isoparms.

You also may also choose to use a Refine OP on the feature before pasting.

ALONG VECTOR SUB-PAGE

projection axis

The axis along which the four corners of the feature are projected onto the base. When adding the feature from the outside as a Vector paste.

<i>X / Y / Z</i>	Cartesian axis – X, Y, or Z.
<i>Feature Normal</i>	Along the primitive normal of the feature.
<i>Minimum Distance</i>	Project curve points to their closest places on surface. This is along the vector of minimum distance between the feature’s corners and the base surface. This may or may not coincide with the base surface normal at that point.
<i>User Defined</i>	Enables the fields immediately below to specify the X, Y, and Z components of the vector.

ray tolerance

The precision of the ray projection algorithm. If you’re not satisfied with the point of intersection the vector projection provides, a lower value may yield a better result.

PARAMETRICALLY SUB-PAGE

u/v range

Enter the parametric U and V sizes here to determine the area on the base surface to be extracted – if you’re spawning, or the area onto which to paste the feature if you’re performing a parametric paste.

AS IS SUB-PAGE

Feature location on the base is given by the shape and position of its knots; no express effort is made to fit it into the space of the base. This is particularly useful when repasting a previously unpasted surface (say you want to unpaste it to model in its pasted shape but without the clutter of the entire hierarchy); just remember to enable *Keep Shape* both in the Unpaste and Paste, and preserve unpasted hierarchy in the Unpaste OP.

preserve unpasted shape

Do not change the position or shape of the feature when (re)applying it to the base. Even if the feature does not appear affected, it will react to base surface deformations if the pasting operation has been successful.

85.3 TIPS

CREATING A CONTINUOUS SURFACE

Use the Paste OP to create a continuous surface which covers multiple surfaces and seams by pasting a flat dense NURBS grid over the entire paste hierarchy with a parametric range of $U=0-1$, $V=0-1$. This makes for easier texturing as you avoid the problems associated with texturing over seams. You could also achieve a similar effect by running the geometry through a Convert OP.

85.4 EXAMPLE FILE

Look in: *\$HD/SOPs/Pasting/head.hip* for an example file.

85.5 SEE ALSO

- *Carve OP* p. 486
- *Convert OP* p. 512
- *Trim OP* p. 819
- *Unpaste OP* p. 831

86 PEAK OP

86.1 DESCRIPTION

The Peak operation translates primitives, points, edges or breakpoints along each of their normals. If no group is specified, all the points are translated along their normals.

When using primitives, the points are translated along the average normal of their primitives. This differs from translating along the points' normal because only primitives specified in the group are considered and the points normal attribute is ignored.

86.2 PARAMETERS

GROUP

Primitives, points, edges or breakpoints to translate.

GROUP TYPE

The type of elements referenced in the group field.

DISTANCE */dist*

The translation distance.

RECOMPUTE POINT NORMALS

Recomputes point normals if they exist.

86.3 LOCAL VARIABLES

NPT	The number of points.
NPR	The number of primitives.

86.4 SEE ALSO

- *Edit OP* p. 553
- *Soft Peak OP* p. 758
- *Transform OP* - p. 805

87 POINT OP

87.1 DESCRIPTION

This very powerful OP – which is the complement to the Primitive *OP* (see *Primitive OP* p. 697) allows you to get right down into the geometry and manipulate the position, color, texture coordinates, and normals of the points in the Source, and other attributes.

For example, you can create point colouring, or flip the normals of incoming geometry. Using expressions in Position X, Y and Z, you can move any given input point to a new place as defined by the expression with any standard attributes.

87.2 PARAMETERS – STANDARD PAGE

POSITION */tx /ty /tz*

Expressions to translate the XYZ coordinates of a given point can be entered here. The variables to modify here are: *\$TX*, *\$TY* and *\$TZ*.

Simply entering *\$TX* into the Position X field means that the X coordinate of each point that comes in is passed straight through with no modification.

Changing this entry to *\$TX+5* means that the X coordinate of each point that comes in will be displaced by 5 units. This formula can be expanded to produce many useful effects. Transformations can also be effected in the Y and Z fields.

KEEP / ADD / NO WEIGHT */weight*

If you select *Add* from the pop-up menu, enter formulas here to control the values of the point weights here. The variable to modify is: *\$WEIGHT*. Values for the weight of the point can range from 0.0001 to infinity.

KEEP / ADD / NO POINT COLORS

Keep leaves the information from the previous OP in the chain untouched. *No* removes the attribute, while *Add* allow you to enter expressions to modify the RGB colors of a given point. The variables to modify are: *\$CR*, *\$CG* and *\$CB*.

KEEP / ADD / NO ALPHA */alpha*

If you select *Add* from the pop-up menu, enter formulas here to control the alpha value (transparency) of a given point. The value can range from 0.0 - 1.0. A value of zero will make the given point fully transparent. A value of one will make the point fully opaque. The variable to modify is: *\$CA*.

KEEP / ADD / NO NORMAL

/nx /ny /nz

If you select *Add* from the pop-up menu, enter formulas to change a given point normal here. Point normals are directional vectors used by other OPs, such as Turbulence, Facet and Copy. See the *Geometry Types > Attributes > Normals* section for detailed information on normals. The variables to modify are: *\$NX*, *\$NY* and *\$NZ*.

flipping normals

You can flip the point normals of incoming geometry by entering:

-\$NX -\$NY -\$NZ

in the fields with this parameter set to *Add Normals*. This works, because it takes the existing normals (*\$NX*, *\$NY*, *\$NZ*), and inverts them (the preceding -).

KEEP / ADD / NO TEXTURE

/mapu /mapv /mapw

If you select *Add* from the pop-up menu, enter formulas here to control the values of the texture coordinates here. The variables to modify are: *\$MAPU*, *\$MAPV* and *\$MAPW*.

87.3 PARAMETERS – PARTICLE PAGE

KEEP / ADD / NO MASS/DRAW

Retains, adds, or removes mass and drag attributes for points.

mass / drag */mass /drag*

The mass and drag coefficient of the particle.

KEEP / ADD / NO TENSION

Tension affects the elasticity of the edges the point is connected to.

KEEP / ADD / NO SPRING K

The Spring Constant is a well known physical property affecting each point.

KEEP / ADD / NO PARTICLE SCALE

Creates, removes, or ignores particle scale attributes defined in the Particle OP.

particle scale */pscale*

Particle Scale acts as a multiplier for the size of particles. The value of this attribute is multiplied by the size specified in the Particle OP's render attributes to scale each particle.

KEEP / ADD / NO POINT VELOCITY

Retains, adds, or removes the velocity of points.

velocity $/vx /vy /vz$

Defines the magnitude of the particle's velocity in the X, Y and Z directions.

KEEP / ADD / NO UP VECTOR

Creates/Removes the “up” attribute for points. This attribute defines an up vector which is used to fully define the space around a point (for particle instancing or copying geometry).

up vector $/upx /upy /upz$

These are the values for the up vector. The up vector is used in conjunction with the copy template's normals to control the orientation of the copies in the Copy OP or for Particle Instancing (geo Object > *Render* page > *Particle Geo*). The variables to modify are: $\$UPX$, $\$UPY$, $\$UPZ$ – which are the values of the up vector attribute for the first source; and $\$UPX2$, $\$UPY2$, $\$UPZ2$ – which are the values of the up vector attribute for the second source.

KEEP / ADD / NO INSTANCE

Creates/Removes the “instance” attribute for points. This attribute defines an object that is instanced on the point. Following are the rules which govern which object will be instanced on a particle:

- If the “instance” attribute exists, it will be used.
- If the “instance” attribute exists but is not set, the particle geo set in the object's *Render* page is used.
- If the “instance” attribute exists but the string is not a valid object, nothing will be instanced on that particle.
- If the “instance” attribute does not exist, the particle geo in the object's render page is used.

87.4 PARAMETERS – FORCE PAGE

KEEP / ADD / NO RADIUS

Used to modify the distance roll-off effect. The roll-off is:

$$r / (r+d^2)$$

Where r is radius, and d is distance from attractor point.
If no radius is set, no attenuation is performed.

KEEP / ADD / NO FORCE SCALE

Multiplier for total force associated with this attractor point.

Both *Radius* and *Force Scale* will default to 1 if not created as point attributes.

RADIAL / NORMAL / EDGE / DIRECTIONAL FORCE

These four parameters introduce a type of force when created and each has a corresponding multiplier associated with it.

radial force

Force directed towards the attractor point. Positive multipliers are towards while negative are away.

normal force

Force directed along the point normal direction.

Note: A point normal attribute does not have to be present, but if present, it overrides the otherwise computed point normal.

edge force

Only works on primitive face types. The force is directed in the direction of the edge leading from that point. If multiple vertices reference the same point, then the direction is the edge direction of the last primitive referencing the point.

If the face open, then the end point has an edge direction equal to that of the preceding point in that primitive.

Note: When edge forces are added using the Point OP, the force directions are computed in the Point OP itself. Thus, any following transformations do not effect these. If you wish for the edge directions to be transformed as well, all transformations must be done before the Point OP. Only the edge forces function like this.

directional force

An arbitrary directional force, still affected by the distance roll-off function.

87.5 PARAMETERS – CUSTOM PAGE

Here, you can explicitly specify an attribute to affect. The attribute will not be created (use the `AttribCreate` OP for that) but it can be changed.

NUMBER CUSTOM

Allows you to specify how many custom attributes to create.

APPLY ATTRIBUTE # */apply#*

Whether to apply this rule.

NAME # */name#*

Name of the attribute (e.g. “Cd”).

SCALAR VALUE */val#[1-4]*

Value to set scalar attributes.

STRING VALUE */sval#*

Value to set string attributes.

87.6 LOCAL VARIABLES

You can use these variables in the: Delete, Group, Point, Primitive, and Vertex OPs.

PT	The point number
NPT	The total number of points
CEX, CEY, CEZ	The centroid of the OP
TX, TY, TZ	The point position
BBX, BBY, BBZ	The relative position of the point with respect to the bounding box. These variables will always be within 0 to 1.
NX, NY, NZ	The normal of the point
MAPU, MAPV, MAPW	The texture coordinates of the point
CR, CG, CB	Red, green and blue color of the point
CA	The alpha value for the point
UPX, UPY, UPZ	The values of the up vector attribute for the first source.
UPX2, UPY2, UPZ2	The values of the up vector attribute for the second source.
PSCALE	The multiplier for the scale of the particles.
AGE	The number of seconds a particle has been alive.

particle specific variables

MASS	The mass of the point
DRAG	The drag coefficient for the point
LIFE	Percentage of life used for a point/particle. This attribute is created by the Particle OP. The “life” attribute contains two fields: life[0] = time (in secs) that the particle has been alive life[1] = time (in secs) of the particles total life span

DIST	Distance of the point to an intersected surface. This attribute is created by the Ray OP.
VX , VY, VZ	Velocity of the point.
ID	The particle ID associated with the point. This variable is only defined if the point is associated with a particle system that has the particle ID attribute enabled.

OVERRIDING DEFAULT INTERPRETATIONS OF VARIABLES

You can now override the default interpretation of a variable. For example, in the Point SOP, \$CR refers to the point colour, then the vertex, then the primitive, then the detail. If there is a point colour, you can still get the vertex colour by doing \$vtxCr.

Current Prefixes:

- pt* - point
- vtx* - vertex
- prim* - primitive
- det* - detail

87.7 USES / WORKS IN RELATION WITH

- Manipulating the position, color, texture coordinates and normals of points.
- Swapping color channels.
- Swapping or modifying texture coordinates.
- Fetching attributes from another OP.
- Tweaking the orientation of copies in the Copy OP and instanced geometry.

87.8 SEE ALSO

- *Particle OP* p. 653
- *Point OP* p. 667
- *Primitive OP* p. 697
- *Transform OP* - p. 805
- *Vertex OP* p. 852
- Objects > geo Object > *Geo Instance* p. 283

88 POLYCAP OP

88.1 DESCRIPTION

PolyCap is used for building polygons out of boundary edges.

88.2 PARAMETERS

GROUP

A list of boundary edges with which to create caps.

REVERSE CAPS

Reverse the orientation of the created caps.

TRIANGULATE CAPS

Construct the caps using only triangles.

UNIQUE POINTS

Create caps with their own points.

UPDATE POINT NORMALS

Recompute point normals, if they exist.

89 POLYEXTRUDE OP

89.1 DESCRIPTION

PolyExtrude is intended for extruding polygonal faces and edges. A local space is computed for the geometry and a symmetry is maintained between the local spaces of the different entities. The local z-axis is always oriented along the normal of the entity. For example, to extrude one unit along a face's normal, set the Translation z field in the Local tab to 1.

89.2 PARAMETERS – LOCAL PAGE

GROUP

Geometry to extrude.

TRANSFORMATION ORDER *//rst*

Order in which to apply local transformation.

ROTATION ORDER *//xyz*

Order in which to apply local rotations.

TRANSLATION *//tx //ty //tz*

Local xyz translation.

ROTATE *//rx //ry //rz*

Local XYZ rotation.

SCALE *//sx //sy //sz*

Local xyz scale.

INSET

Amount by which to inset the face, on edges this parameter is treated as an addition to x scale./inset.

SYMMETRY AXIS

Sets the local axis of the first face that is used to mirror the extrusion.

89.3 PARAMETERS – GLOBAL PAGE

TRANSFORMATION ORDER */grst*

Order in which to apply global transformation.

ROTATION ORDER */gxyz*

Order in which to apply global rotations.

TRANSLATION */gtx /gty /gtz*

Global XYZ translation.

ROTATE */grx /gry /grz*

Global XYZ rotation.

SCALE */gsx /gsy /gsz*

Global XYZ scale.

89.4 PARAMETERS – OPTIONS PAGE

DIVISIONS */divs*

Number of divisions along the side of extrusion.

OUTPUT FRONT

Toggle whether or not to output a front face, ignored for edges.

OUTPUT BACK

Toggle whether or not to output a back face, ignored for edges.

OUTPUT SIDE

Toggle whether or not to output side faces.

REMOVE ZERO AREA SIDES

When enabled, sides that are degenerate are removed. Such sides can result from an extrusion that scales but does not translate.

REMOVE SHARED SIDES

When enabled, extruded sides that lie one on another are removed. Such sides can result from global extrusion of adjacent faces.

CONSOLIDATE POINTS

<i>None</i>	No points are consolidated.
<i>Per-Face</i>	Points created by each face are consolidated.
<i>All</i>	All new points are consolidated.

89.5 PARAMETERS – GROUP PAGE

OUTPUT GROUPS

When on, primitives groups are exported.

FRONT GROUP

Name of group in which to place front faces.

BACK GROUP

Name of group in which to place back faces.

SIDE GROUP

Name of group in which to place side faces.

89.6 SEE ALSO

- *Extrude OP* p. 563

90 POLYKNIT OP

90.1 DESCRIPTION

PolyKnit can be used to manually knit joining polygons between existing polygons. Polygons are created by specifying a list of input points from which to "knit" the new polygons. Both triangles and quadrilaterals can be created. This tool allows you to specify a minimum number of points along which it can find the points to knit. For example, with only four points, PolyKnit can find two paths of unshared connected edges and use their points to knit polygons. Each of the paths is called a "meta" edge. The polygons that are created along these paths can either be triangles (T) or quadrilaterals (Q). The paths are computed to minimize the number of edges that are produced (and thus minimizing the number of polygons that are produced).

It is important to note that points are specified in a strip-like fashion. The points should be specified in an order such that they zig-zag along the geometry to knit. The first two points, for example, should each belong to the opposite geometries that are to be knitted together.

EXAMPLES

0 1 2 3 4	Builds 3 triangles (0, 1, 2), (1, 2, 3), (2, 3, 4).
0 1 2 q 3 4	Builds 1 triangle and 1 quad (0, 1, 2), (1, 2, 3, 4).
T 0 1 2	Builds triangles using point 0, and the points along the shortest path from point 1 to point 2. Points 1 and 2 must be connected.
Q 0 1 2 3	Builds quads using the points from the shortest path connecting points 0 and 3, and 1 and 2.

90.2 PARAMETERS

POINT LIST

Specifies the list of points which to knit.
The following special characters can be used:

t	Currently building triangles.
q	Currently building quadrilaterals.
Q	Currently building "meta" quadrilaterals.
T	Currently building "meta" triangles.

COLLAPSE QUADS

Remove extra vertex from quadrilaterals that are actually triangles.

Remove Degenerate Polygons - Do not generate degenerate polygons (those collapsing to a single point or edge).

Flip Polygon Normals - Reverse order of vertices.

Unique Points - Do not reuse points from the input geometry. Only applicable if Keep Original is on.

Update Point Normals - Recomputes point normals if they exist.

Keep Original - Keep the input geometry.

See also: PolyCap, PolyLoft, PolyStitch

91 POLYLOFT OP

91.1 DESCRIPTION

The Polyloft OP generates meshes of triangles by connecting (i.e. lofting/stitching) the points of open or closed faces without adding any new points. Polyloft can also connect groups of unrelated points in a similar fashion. The faces and the point groups need not have the same number of points.

The optional input specifies the rest geometry, typically the copy of the main input at a specific face (usually 1). This forces the point order to remain constant throughout the animation and prevents the triangular stitch from popping as the geometry deforms. If you specify face or point groups for lofting using rest geometry, make sure they are defined in the second input.

91.2 PARAMETERS

CONNECT CLOSEST ENDS

Start stitching at the two closest points, and handle arbitrary face orientation and start vertices.

CONSOLIDATE POINTS

Fuse neighbouring points before stitching.

DISTANCE */dist*

Threshold distance for consolidation.

MINIMIZE

Distance minimization goal:

2-Point Distance Default stitching target.

3-Point Distance May help avoid intersections.

U WRAP

Close the stitch in U (close each cross-section)

V WRAP

Connect first and last cross-sections

CREATE POLYGON GROUP

Place the generated triangles into a group.

FACES TAB

Stitch a set of faces by connecting their control vertices.

group

Subset of faces to loft.

keep primitives

Preserve the cross-sections after stitching.

POINTS TAB

Stitch up to 6 sets of points, each set acting as a cross-section.

group 0-5

Point group.

91.3 SEE ALSO

- *Join OP* - p. 602
- *PolyPatch OP* p. 681
- *PolySpline OP* p. 685
- *Skin OP* p. 749
- *Stitch OP* p. 775

92 POLYPATCH OP

92.1 DESCRIPTION

The PolyPatch OP creates a smooth polygonal patch from a mesh primitive or a set of faces (polygons, NURBS or Bezier curves).

This OP reproduces *action's* Patch SOP.

92.2 PARAMETERS

GROUP

Subset of input to use. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

BASIS

Select spline type: Cardinal or BSpline.

CONNECTIVITY

<i>Rows</i>	Creates horizontal lines, which are open polygons.
<i>Columns</i>	Creates vertical lines, which are open polygons.
<i>Rows & Cols</i>	Both rows and columns, all open polygons.
<i>Triangles</i>	Build the grid with triangles.
<i>Quadrilaterals</i>	Default grid connection – four-sided quadrilaterals.
<i>Alternating Triangles</i>	Generates triangles that are opposed; similar to the <i>Triangles</i> option.
<i>Inherit from Source</i>	When this connectivity option is selected, the output mesh will have the same connectivity as the input hull mesh.

U WRAP / V WRAP

<i>Off</i>	Do not wrap in U/V.
<i>On</i>	Wrap in U/V.
<i>If the primitive does</i>	Wrap in U/V if the input primitive does.

U CLAMP (FIRST) / U CLAMP (LAST)
V CLAMP (FIRST) / V CLAMP (LAST)

<i>Off</i>	Do not clamp the first/last end in the U/V direction.
<i>On</i>	Clamp the first/last end in the U/V direction.
<i>If primitive does</i>	Clamp the first/last end in the U/V direction if the input primitive does.

OUTPUT DIVISIONS */divisionsx /divisionsy*

The number of divisions in the output surface. Use more divisions for a smoother surface.

OUTPUT POLYGONS

Force polygonal rather than mesh output.

92.3 LOCAL VARIABLES

none.

92.4 SEE ALSO

- *PolyLoft OP* p. 679
- *PolySpline OP* p. 685

93 POLYREDUCE OP

93.1 DESCRIPTION

The Polyreduce OP reduces a high detail polygonal model into one consisting of fewer polygons. The second input's polygons represent feature edges. They are matched to the input mesh by point numbers.

The methods to reduce polygonal models are:

- 1) A percentage of their former size
- 2) A specific number of polys (within a few)
- 3) According to distance from a camera

Note that as it requires (and outputs) a triangular mesh, the polygon count may increase as a result of this operation.

A second input for feature edges is provided.

93.2 PARAMETERS

POLYGONS */reduce*

The polygons which will be candidates for simplification. Other polygons which share points with these might also be affected.

FEATURES */creates*

Which polygons are feature edges.

PERCENTAGE TAB

Choose reduction level with a percentage.

keep % */percentage*

The percent of polygons to keep. This is with respect to the triangulated mesh.

NUMBER OF POLYGONS TAB

Specify a desired number of polygons

keep # */numpolys*

The desired number of triangles in the simplified model The resulting number may be slightly less than this.

DISTANCE TAB

Reduce polygons based on distance to an object.

object */obj*

The object to use as a reference.

dist. threshold */distance*

The world distance at which the polygons should be left at full detail.

minimum % */minpercent*

A lower bound to the level of reduction.

STIFFEN BORDER */borderweight*

Without any constraints, the edges of planar surfaces can erode. This controls a bias which penalizes such erosion.

STIFFEN FEATURES */creaseweight*

The amount of penalty to add to the feature edges being eroded.

EQUALIZE EDGES */lengthweight*

This bias penalizes the removal of long edges. It tends to reduce high aspect ratio triangles at the expense of more uniform reduction.

PREVENT MESH INVERSION */meshinvert*

When enabled, each reduction is tested to see if it would flip a triangle normal. While incurring a slight cost, the results are almost always worth it.

PRE-TRIANGULATE */triangulate*

As only triangular polygons will be reduced, this option will automatically triangulate the input polygons.

PREVENT CRACKING */keepedges*

This prohibits the removal of any edge that occurs at the boundary of the polygons. This ensures no cracks develop with unreduced areas.

93.3 SEE ALSO

- *Stand Alone Tools* section > greduce

94 POLYSPLINE OP

94.1 DESCRIPTION

The Polyspline OP fits a spline curve to a polygon or hull and outputs a polygonal approximation of that spline. You can choose either to create divisions between the original points, or to ignore the position of the original points and divide the shape into segments of equal lengths.

Polyspline can optionally resample the output curve, providing control over the length and number of its segments.

Tip: When using this OP, it is useful to enable Points display in the Viewport options dialog. This way you can see exactly what effect the OP is having.

Note for action users: The Polyspline OP is the equivalent of *action*'s Spline OP.

94.2 PARAMETERS

GROUP

Subset of faces to use. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

SPLINE TYPE

Spline type to use. There are seven choices:

<i>Bspline</i>	Softer curve – does not pass through the original points of the Source.
<i>Cardinal</i>	The curve passes through the original points of the Source.
<i>Linear</i>	Straight line segments.
<i>Bezier,</i> <i>Degree 2 Bezier,</i> <i>Special Bezier,</i> <i>Special Smooth Bezier</i>	Variations on Beziers.

CLOSE

<i>Off</i>	Output spline open.
<i>On</i>	Output spline closed.
<i>If polygon does</i>	Use the closure of the input face. In other words, Closed splines are created if the Source polygons are closed, open splines are created if the Source polygons

are open.

DIVISION METHOD

<i>Standard</i>	Do not resample curve.
<i>Even length segments</i>	Resample - ensure equal segment lengths.
<i>Even X Segments</i>	Resample - ensure segments have equal length in X.
<i>Even Y Segments</i>	Resample - ensure segments have equal length in Y.
<i>Even Z Segments</i>	Resample - ensure segments have equal length in Z.

SEGMENT LENGTH */segsize*

The length of the segments in the resampled curve.

If *Division Method > Standard* is selected, this has no effect.

If *Even Length Segments* are selected, *Segment Length* sets the length of output segments. The number of output segments is determined by *Output Divisions*.

If *Output Divisions* is zero, the number of output segments is calculated using the *Segment Length* parameter, and is determined by how many segments of this size will fit into the overall shape.

If *Even Length Segments* is selected, along with zero *Output Divisions* and *Segment Length* of zero, an error message is generated, saying “Invalid number of divisions or segment size”.

OUTPUT DIVISIONS */polydivs*

Number of segments in the resampled curve.

If *Division Method > Standard* is selected, this has no effect. If *Even Length Segments* is selected, this parameter sets the number of edges that is created. The length of the segments is determined by *Segment Length*. If *Segment Length* is 0, the length of the output segments is determined by dividing the over all shape into this number of segments.

If the *Output Divisions* parameter is set to zero, the value of the *Segment Length* parameter is used to calculate the number of Output Divisions.

SAMPLE DIVISIONS */legdedivs*

Number of spline divisions before resampling.

If *Division Method > Standard* is selected, this is the number of subdivisions for every edge. If *Even Length Segments* is chosen, it has the subtle effect of determining the accuracy with which the segment lengths can be calculated.

FIRST CV COUNT */first*

Number of times to repeat the first control vertex, determining its multiplicity. This determines the number of times to replicate the first vertex of the Source polygon(s). This is most useful when the Source consists of open polygons; extra vertices at the beginning of the line will force the curve to extend to the beginning of the line. For example a value of two will force a Cardinal curve to extend to its first vertex and a value of three will force a bspline to start at its first vertex.

LAST CV COUNT */last*

This determines the number of times to repeat the last control vertex, determining its multiplicity. This determines the number of times to replicate the last vertex of the Source polygon(s). This is most useful when the Source consists of open polygons; extra vertices at the end of the line will force the curve to extend to the end of the line.

CV TENSION */tension*

The tension exerted by the points from the Source polygons. The greater the tension, the closer the resulting shape will be to the original shape.

94.3 SEE ALSO

- *PolyLoft OP* p. 679
- *PolyPatch OP* p. 681



95 POLYSPLIT OP

95.1 DESCRIPTION

PolySplit can be used to cut polygons. It can handle a chain of cuts, possible self-intersecting, as well as cuts between non-adjacent polygons.

Cuts are made by specifying locations on polygons. Locations can be edges or vertices and do not necessarily have to be within the same polygon, nor do they have to be between two adjacent polygons. If a cut is made from one polygon to another non-adjacent polygon, the operation will try to cut across all the polygons in between. If no connected path exists along the cut, the operation will cut as far as it can from either end.

The new polygons that are created preserve vertex and primitive attributes of the polygon from which they were cut. Group memberships are also preserved.

A cut made from one end vertex to another end vertex of an open polygon will close the polygon.

There are two algorithms that PolySplit can use to find the path from the source to the destination. The 'Shortest Distance' algorithm tries to find the shortest path through connecting polygons. The 'Quad Strips' algorithm finds a path through quads, and moves from one quad to the next by crossing opposite edges. The latter algorithm cannot be used to cut through vertices, and self-intersecting cuts may give undesired results. Unlike 'Shortest Distance', 'Quad Strips' will not attempt a cut unless a complete path can be found.

If only one split location is specified and both 'Quad Strips' and 'Close Path' are used, PolySplit will try to find a closed loop around the quad strip.

95.2 PARAMETERS

SPLIT LOCATIONS

Specifies the chain of locations along which to make cuts. The list of locations is separated by a space and can be specified in the following ways:

<i>avb</i>	Cut at vertex <i>b</i> of primitive <i>a</i> where <i>a</i> and <i>b</i> are integers (e.g. 0v3).
<i>aeb</i>	Cut at the midpoint of edge <i>b</i> of primitive <i>a</i> where <i>a</i> and <i>b</i> are integers (e.g. 0e3).
<i>aeb:t</i>	Cut at <i>t</i> percent along the edge <i>b</i> of primitive <i>a</i> where <i>a</i> and <i>b</i> are integers and <i>t</i> is a real number between 0.0 and 1.0 (e.g. 0e3:0.7 cuts at 70% along the edge 3 of primitive 0).
<i>pa-b</i>	Cut at the midpoint of the edge between points <i>a</i> and <i>b</i> , where <i>a</i> and <i>b</i> are integers (e.g. p1-3).

$pa-b:t$

Cut at t percent along the edge between points a and b , where a and b are integers (e.g. p1-3:0.7).

PATH TYPE

Specifies the algorithm (see *Description*, above) used to find the desired path.

OVERRIDE BIAS

Specifies an overriding value for cut locations made along edges, even for those where the midpoint is used by default.

UPDATE POINT NORMALS

Update point normals if they exist.

CLOSE PATH

Causes an extra cut to be made from the last split location to the first.

95.3 SEE ALSO

- *SubDivide OP* - p. 777

96 POLYSTITCH OP

96.1 DESCRIPTION

The Polystitch OP attempts to stitch polygonal surfaces together, thereby eliminating cracks that result from evaluating the surfaces at differing levels of detail.

First, the boundaries of all the polygons to be stitched are found. An edge is a boundary edge if it is shared by no other polygon. The uniqueness of edges is determined by point numbers, and not by spatial positioning. Each boundary is then split at each “corner” into a number of pieces. A list of corner points can be manually specified, or any point at which the boundary changes direction by a certain amount can be flagged as a corner.

Finally, any two boundary pieces that are within the tolerance of each other are stitched together. This is performed by snapping the points of the high detail edge to those of the low detail edge.

96.2 PARAMETERS

GROUP

Subset of faces to use. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

POLYGONS TO STITCH

The polygons to consider for stitching.

CORNER POINTS

A list of point numbers that are to be considered breaks in the boundary edges.

MAX DIST TO STITCH */tol3d*

The maximum distance two edges can be from each other and still be stitched.

CONSOLIDATE POINTS */consolidate*

When several points along *one* edge are snapped to the same position, consolidate them into a single point. This only consolidates along the boundary, not across the boundary.

AUTOMATICALLY FIND CORNERS */findcorner*

Whenever an edge changes direction at a point more than the specified angle, it will mark that point as a corner.

CORNER ANGLE */angle*

The maximum angle a boundary point can change before it is considered a corner.

96.3 SEE ALSO

- *PolyLoft OP* p. 679

97 POLYWIRE OP

97.1 DESCRIPTION

PolyWire constructs a polygonal wire around a polygonal backbone. The four numerical parameters support all the local variables of the Point operation, plus the LSYSTEM specific variables of \$WIDTH, \$SEGS, \$DIV, \$LAGE, \$GEN, and \$ARC.

97.2 PARAMATERS

GROUP

The group of polygons to convert to wires.

WIRE RADIUS */radius*

This is the radius of the wire to sweep over the polygon.

MAXIMUM JOINT SCALE */maxscale*

Prevent Joint Buckling scales up the intersection points so that they lie on the intersection of two tubes, rather than a width sized sphere. This can cause the points to be scaled past the first segment of the tube, however, causing buckling. This parameter allows you to change on a point level what the maximum scale applied to the points is.

SMOOTH POINT */smooth*

This is an integer value. If true for a point, the wire will intersect at that point with smooth manifold topology. If zero, it is just capped.

DIVISIONS */div*

This is the number of divisions in the circle which is to be swept over the polygon. It can vary on a point basis.

SEGMENTS */segs*

The number of segments to divide each edge of the polygon into. It can vary on a point basis.

SEGMENT SCALES */segyscale*

These are how far into the segment to make the first circle and how far towards the end to keep going. Both values are in the range 0-1, where 0 is the start and 1 the end. These are segment specific values.

The local var \$NSEG is available here for the number of segments in the previous parameter.

PREVENT JOINT BUCKLING

This toggle when set will scale the intersection point of edges to avoid a collapse when sharp turns are made.

DO VERTEX TEXTURES

Toggles whether vertex textures are generated for the geometry. For best texturing of branch points, the points of the wire should be ordered so the 'important' side has least point number. For example, in a Y branch, the bottom of the Y should have the smallest point numbers.

U SEAM OFFSET */uoff*

A per segment parameter which cycles how far around the tube the seam is placed. The seam is always snapped to the nearest polygon edge.

U TEXTURES */textu*

Per segment values for the starting/ending values of the u texture.

V TEXTURES */textv*

The same, but for v textures.

JOINT UP VECTOR */upvector*

If enabled, the up vector at each joint is set to the specified value. This will result in twisting of the branches.

97.3 SEE ALSO

- *L-system OP* p. 618
- *Wireframe OP* p. 859

98 POP OP

98.1 DESCRIPTION



Use a Pop OP to get the geometry from a POP network back into your scene.

The Pop OP uses a POP network to cook the particle simulation. By itself, the POP network merely describes how the particles should behave. This OP is needed to actually provide geometry for the POP network to cook. The POP OP's cooked geometry will be a particle system that can then be used in your scene.

Tip: If you click on the POP OP's tile icon, there will be an additional item in the pop-up menu: *Edit POP Network...* which brings you to the POP Editor and will also set the View Operation parameters of the POP Editor to match those in the POP OP.

It is important to note that the View Operation in the POP Editor > View Operation ⇨ Dialog will ignore the settings in all POP OPs. However, the POP network will adopt the values of any Pop OP when you use that specific Pop OP's option to directly enter a POP network using its pop-up menu.

CONTEXT GEOMETRY

All POP networks can have 'Context Geometry'. These context geometries are set in the parameters of the POPNET, the Particle CHOP, or the POP OP. In the POP OP they can also be set by wiring OPs into the inputs of the POP OP. These context geometries can be referenced by the following POPs: Source POP, Collision POP, SoftBody POP, Attractor POP, Creep POP (All of which have the parameter - Geometry Source > *Use Context Geometry*). So, instead of hard-coding a specific piece of source geometry or collision geometry, a single POP network can be used by several POP OPs to generate several different outputs, depending on the geometry input into the POP OP.

98.2 PARAMETERS

POP NETWORK / POP NAME

Specify the Particle Operator and network here.

START TIME */timestart*

The start time is the time at which the POP simulation starts cooking. Before this, the particle system will be empty. The default is *Tstart* (frame one).

PREROLL TIME */timepreroll*

How many seconds of the POP simulation to bypass, after the reset time is reached. For example, if you put the number 33 into this field (and reset is at *Tstart*), frame one will show the simulation that was at a time of 33 seconds. In other words, the

first thirty-two seconds have been bypassed, and the time at thirty-three seconds is shifted to frame one. The first thirty-two seconds must still be calculated in order to compute the status of the points, so you will notice some delay upon reset.

INITIAL STATE

Instead of starting from an empty particle system, you can use a geometry file to specify the initial state for the simulation. Create a geometry file by using a Geometry Output OP, and specifying a POP OP as its output.

OVERSAMPLING

Oversampling is how many times in between frames to cook the simulation. For example, a value of 1 means to cook once per frame. A value of 2 means to cook at frame 1, frame 1.5, frame 2, etc.

Note: The Sampling Rate in the POP Editor is not associated in any way with the oversampling in the OP. The OP's oversampling is the only one used when rendering, or for display of the OP's data. The POP Viewport oversampling is only for the POP editor (there may be multiple ops referencing the same POP network with different oversampling rates).

MAXIMUM # PARTICLES */maxparticles*

Controls the maximum number of particles that can exist in the simulation at any given moment. A value of 0 means particles can always be birthed.

REMOVE UNUSED POINTS

Removes all unused points from the input geometry. This is provided as an explicit option instead of happening automatically because it saves the time needed to purge the points from memory during the simulation. This prevents unnecessary slow-downs during the simulation.

98.3 PARAMETERS – RBD PAGE

REST THRESHOLD */restthreshold*

The minimum relative velocity two RBD particles must have in order to be considered colliding.

CONTACT TOLERANCE */contacttol*

The distance at which RBD particles are treated as being in contact.

MAX TIME SPLITS */maxdivisions*

A measure of how accurately RBD collisions should be detected.

CONSTRAINT TIME */constrainttime*

Approximate time allowed to satisfy RBD constraints.

SOLVER TYPE

Type of solver to be used by the RBD solver.

98.4 SEE ALSO

- *Particle OP* p. 653
- *Ref > Particles (POPs)* section

99 PRIMITIVE OP

99.1 DESCRIPTION



This OP is much like the Point OP, and is useful for controlling a primitive's position, size, orientation, color, alpha, in addition to primitive-specific attributes, such as reversing primitive normals (*Face/Hull Page > Vertex > Reverse*).

You can also apply parametric affine transformations to a profile by using this OP. You can also use it to open, close, reverse, and cycle the profile curves.

Note: When applying transformations to a profile, you can only rotate about the Z axis because the projected curve is a planar curve that lives in the domain of the surface. Therefore it wouldn't make any sense to allow rotations in X or Y for profiles.

TRANSFORMATION OF PRIMITIVES VS PROFILES

A Bézier surface is a single primitive, as is a NURBS surface, while a polygon mesh can consist of hundreds of individual primitives. Care must be taken to ensure the desired result. Profiles can be translated, rotated, and scaled along with 3D primitives. The Z component of translation and scaling is ignored. The X and Y components would be interpreted as U and V values because they apply to the space in which profiles are defined.

EXAMPLE – MAPPING A TEXTURE INSIDE A SPHERE

There are many uses for the Primitive OP. Normally, if you apply a texture onto a sphere, it is mapped onto the outside surface because the U surface normals point outwards by default. If you wanted to map the texture onto the inside of the sphere instead, you could simply run the sphere geometry through a Primitive OP, and select Reverse U (i.e. the surface normals) in the *Face/Hull* page > *Vertex* menu.

99.2 PARAMETERS – TRANSFORM PAGE

SOURCE GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. You can specify profile curves within the group by providing a profile pattern (e.g. *.3 specifies the fourth profile in all spline surfaces).

Tip: By specifying both a primitive and a profile here (example: 0 0.*), you can affect a transformation of *both* the parent surface and a profile curve.

TEMPLATE GROUP

A subset of template points to transform to.

DO TRANSFORMATION

When checked, allows transformations to occur.

ROTATE TO TEMPLATE

A template can be specified using the second input of the Primitive OP. When set to *On*, this template can be used to transform each primitive to the location and orientation of the template point. This is similar to what the Copy OP does except that the actual primitives are transformed, not copies made.

- Off* Don't rotate.
- On* The primitive gets rotated as if its normal was (0,0,1), and is meant to point the same direction as the template normal.
- Match Normals* Rotates the primitive so that its real normal lines up with the normal of the template point.

Tip: This option is very useful when combined with POPs where particles can be birthed from the centre of a primitive. The particle system can be used as the template so that the primitive is controlled by the motion of the particle system.

TRANSFORM ORDER

Sets the overall transform order for the transformations. The transform order determines the order in which transformations take place. Depending on the order, you can achieve different results using the exact same values. Choose the appropriate order from the menu.

ROTATE ORDER

Sets the order of the rotations within the overall transform order.

TRANSLATE */tx /ty /tz*

These three fields move the input geometry in the three axes. Profiles use *tx* and *ty* only.

ROTATION */rx /ry /rz*

These three fields rotate the Source geometry in the three axes. Profiles used *rz* only.

SCALE */sx /sy /sz*

These three fields scale the Source geometry in the three axes. Profiles use *sx* and *sy* only.

PIVOT */px /py /pz*

The pivot point for the transformations. Profiles use *px* and *py* only.

LOOKAT OBJECT

Selects the object the primitive should point towards. This performs the lookat in object space; if you need to a lookat in world space, use the lookat in the object's *Transform* page instead.

Tip: In order to get multiple sprites to always be perpendicular to the camera, feed them into a Primitive OP and specify the camera as your *Lookat* object. Then the sprites should always be perpendicular to the camera.

UP VECTOR

/upvectorx /upvectory /upvectorz

Defines the orientation of the primitive along the X, Y, or Z axes.

The *Up Vector* determines how a primitive orients itself with respect to the target object (specified in *Lookat*). The default value is of 1 in the Y direction. This will produce nice behaviour if you want the object to rotate somewhat in the Z axis as the *Lookat* object gets very close and/or passes the target. Scaling the up-vector will cause the *Lookat* primitives to remain more upright as they get very close and/or pass the target. The stronger the up-vector, the more the primitives will want to stay vertical and resist the rotation.

99.3 PARAMETERS – ATTRIBUTES PAGE**KEEP / ADD / NO COLORS**

If *Keep* is selected, the color attribute is left unchanged. If *Add* is selected, this parameter changes the color of input primitives according to diffuse color field. If *No* is selected, the color attribute is removed.

KEEP / ADD / NO ALPHA */alpha*

If *Keep* is selected, the alpha attribute is left unchanged. If *Add* is selected, this parameter generates an alpha channel for the primitive. If *No* is selected, the alpha attribute is removed.

KEEP / ADD / NO CREASE */crease*

If *Keep* is selected, the crease attribute is left unchanged. If *Add* is selected, this parameter generates a crease attribute for the primitive. If *no* is selected, the crease attribute is removed.

The “Crease Weight” attribute is used to set edge crease weights for subdivision surfaces (see *SubDivide OP* - p. 777). The crease weight attribute for a primitive sets all edges of the polygon to the value specified. On shared edges, the maximum of the two crease weights is used to define the sharpness of the subdivided surface. Crease weights should be larger than 0, with larger values defining sharper edges.

KEEP / ADD / NO TEXTURE */texture*

If *Keep* is selected, the texture attribute is left unchanged. If *Add* is selected, this parameter generates a texture attribute for the primitive. If *no* is selected, the texture attribute is removed.

This attribute is not used by Houdini, but can be passed down to renderers which support this attribute (i.e. vmantra, RenderMan).

99.4 PARAMETERS – FACE / HULL PAGE**PRESERVE SHAPE U / V**

These options only become available once a type of clamping or closure has been selected.

CLOSURE

Change the closure and clamping of a face or hull. The options are:

close u / v

Close the primitive in U / V. Select from: *Open*, *Closed Straight*, *Close Rounded*, and *Unroll*. When you unroll a closed curve you duplicate the wrapped points (you make them unique) and turn the curve into an open curve. The shape of the curve does not change. Same goes for hulls, only there we unique entire rows and cols. This addresses a problem with texturing wrapped surfaces whereby the texture repeats itself in the wrapped portion of the surface.

clamp u / v

Clamp the primitive in U / V. Select from: *Clamp*, *Unclamp*.

VERTEX

Tip: To reverse the surface normals, then select *Reverse U*. This would have the effect of – for example – projecting texture maps on the inside of a sphere instead of outside (as is the default).

No Change

Does not affect the ordering of the vertices.

Reverse

Reverses both U and V for hulls, and just U for faces. For example, say you had a closed polygon with vertices 0 to 6 that wraps back to 0. When when reversed, vertex 0 remains as the starting point, and only then comes vertex 6. Previous releases of Houdini would have made vertex number six the first in sequence.

Reverse U

Reverses column order of hulls.

Reverse V

Reverses row order of hulls.

Swap U and V

Interchanges rows/columns while preserving topology.

Shift Cycles the vertices by “U Offset” and “V Offset”.

U / V OFFSET */vtxuoff /vtxvoff*

Cycles face or hull columns / rows when the *Shift* operation is selected.

99.5 PARAMETERS – META PAGE

META-SURFACE WEIGHT

When selected, allows meta-surface weighting.

WEIGHT */metaweight*

Enter weight of meta-surface here when *Meta-surface Weight* is selected.

99.6 USES / WORKS IN RELATION WITH

- Opening and closing / clamping and unclamping faces and hulls.
- Reversing primitives that are facing the opposite direction as adjacent primitives.
- Modifying primitive colors.
- Applying transforms per-primitive.

99.7 LOCAL VARIABLES

99.8 LOCAL VARIABLES

You can use all local variables of the Point OP – see *Local Variables* p. 671.
Here is a subset:

N	The number of incoming primitives / points.
PT	The Point number.
PR	The Primitive number.
NPT	The total number of points.
NPR	Total number of primitives
TX, TY, TZ	The point position.
CEX, CEY, CEZ	The centroid of the primitive
DX, DY, DZ	The direction from the centroid of the OP to the centroid of the primitive.

Primitive OP

NX, NY, NZ	The normal of the primitive. Will be 0 for some primitives like sphere and tube (which don't have normals).
CR, CG, CB	Red, green and blue primitive color.
CA	Primitive alpha color.
WEIGHT	The current weight of any meta primitive. For non-meta primitives, this will be 0.

99.9 SEE ALSO

- *Copy OP* p. 519

I00 PRIMITIVE SPLIT OP

I00.1 DESCRIPTION

For each point in the input geometry, Primitive Split tests the specified attribute to see if it is within tolerance on each primitive sharing that point. For each group of vertices outside the tolerance, a new point will be created. This operation is useful for splitting the topology of a polygon mesh for trisstripping or other operations.

I00.2 PARAMATERS

GROUP

The point group to act on.

ATTRIBUTE

The name of the primitive attribute to perform the test on.

TOLERANCE */tol*

The maximum error in any dimmension of the attribute before it will be split.

I00.3 SEE ALSO

- *Vertex Split OP* p. 855

101 PROFILE OP

101.1 DESCRIPTION

The Profile OP enables the extraction and manipulation of profiles.

You will usually need a Trim, Bridge, or Profile OP after a Project OP. Use a Trim OP to cut a hole in the projected surface; use a Bridge OP to skin the profile curve to another profile curve.

101.2 PARAMETERS

GROUP

This field allows you to specify the particular group of curves on the surface. Other primitives are ignored. You can specify profile curves within the group by providing a profile pattern (e.g. *.3 specifies the fourth profile in all spline surfaces).

EXTRACT

These parameters allow you to extract a stand-alone 3D curve as the world or parametric image of the profile. The non-parametric option will yield a curve whose shape and position in space are identical or very similar to those of the chosen profile. The parametric option will produce a planar, XY face whose vertices and type will be identical to those of the profile in 2D; also, if the profile is a spline it will have the same basis as the extracted curve.

parametrically to xy

This parameter tells Houdini whether to extract the profile parametrically or through fitting.

If Fitted (this option not checked), the profile will be a spatial NURBS curve whose position and shape in space will be identical to the curve on surface. It may very well be non-planar.

If Extracted Parametrically (this option checked), the result will extract the profile's parametric image as a 3D face, which will be a planar face in XY whose type (polygon, Bézier, NURBS) will be the same as the profile's, and whose vertices match the profile CVs. This, however, does not guarantee spatial coincidence between profile and extracted curve, and is more of an analytical tool.

Tip: A profile extracted parametrically can be reapplied to the surface identically with the Project OP by choosing the Parametric option with no Mapping to Range. Use this method to extract the profile, pull its points or edit it as you would any 3D face, then re-project it on the surface at the same location.

See also the comments in the *Project OP > Map Profile to Range* p. 710.

smooth curve

If enabled, it will fit a spline through the extracted points. This parameter is disabled when extracting the profile parametrically. Disable this parameter in order to bypass the fitting process which might be both expensive (in processing speed) and unnecessary when extracting profiles produced by boolean operations (see *Surfsect OP* p. 788).

divisions per span */sdivs*

The number of points per span to be computed on the profile. A span is the line connecting two consecutive CVs on a polygon, or the arc between two breakpoints on a spline curve. The profile tends to become more accurate as the number of divisions is higher.

tolerance */tolerance*

This parameter specifies the precision of the fitting process for the extracted 3D data, and is typically less than 0.01.

order */order*

The spline order of the resulting 3D curve. The type of curve (Bézier or NURBS) is inherited from the spatial curve. The order, however, is not inherited because the spline order provides useful control over the quality of the fit. If the profile is a polygon, the spatial curve will be a NURBS curve.

preserve sharp corners

Controls the precision with which sharp corners in the profile curve are interpolated. It should be on when the profile has areas of high changes in curvature.

keep surface

Specifies whether the parent surface should be removed after the extraction or not.

delete original profile

Removes the profile from the surface that owns it.
This option is available only if *Keep Surface* parameter is on.

REMAP

Repositions and scales an existing profile to fit within the specified domain range. It is a good means of bringing an invisible profile into view by setting the mapping range between 0 and 1 in the U and V parametric directions. A profile mapped outside the unit domain becomes invisible but is not removed from the surface. Other ways to change a profile are available through the Primitive OP.

mapping type

Uniform Converts the spatial coordinates of the profile to (U,V) points in the domain of the surface without taking into account the parameterization of the surface.

Chord Length

Takes into account the surface parameterization and attempts to compute a profile best suited to the spatial and parametric determinants of the model.

u range

/urange1 /urange2

Indicates in percentages what part of the U surface domain is the mapping area. A full range of 0-1 will cause the profiles to be mapped to the entire domain in the U parametric direction. The range is not restricted to the 0-1 interval.

v range

/vrange1 /vrange2

Indicates in percentages what part of the V surface domain is the mapping area. A full range of 0-1 will cause the profiles to be mapped to the entire domain in the V parametric direction. The range is not restricted to the 0-1 interval.

101.3 SEE ALSO

- *Primitive OP* p. 697
- *Project OP* p. 707

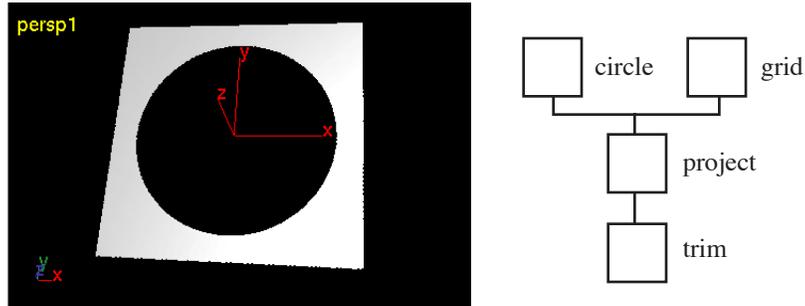
I02 PROJECT OP

I02.1 DESCRIPTION

The Project OP creates curves on surface (also known as trim or profile curves) by projecting a 3D face onto a spline surface, much like a light casts a 2D shadow onto a 3D surface. There are two projection methods: along a vector, or by mapping the face directly onto the parametric space of the surface.

You will usually need a Trim, Bridge, or Profile OP after a Project OP.

For example, in the case of a Trim OP we might have:



- Use a Trim OP to cut a hole in the projected surface (as shown above).
- Use a Bridge OP to skin the profile curve to another profile curve.
- Use a Profile OP to extract the curve on surface or remap its position.

If you end up with a profile curve that is not visible, it may still exist. Confirm a profile curve's existence by clicking on the OP's info pop-up.

ADDITIONAL OPERATIONS FOR PROFILE CURVES

To delete a projected curve, use a Delete OP, and enter the profile number (e.g. 1.4 returns the fifth profile on the second primitive (counting starts at 0)). You can visualise the number of the profiles by enabling the *Profile Number* icon in the Viewport Display options.

You can group the profile curves with a Group OP. Do this by typing the profile numbers in the *Pattern* field. You can use all regular expressions.

You can apply parametric affine transformations to the profile by using a Primitive OP. You can also use the Primitive OP to open, close, reverse, and cycle the profile curves.

Note: When applying transformations to a profile in the Primitive OP, you can only rotate about the Z axis because the projected curve is a planar curve that lives in the domain of the surface. Therefore it wouldn't make any sense to allow rotations in X or Y for profiles.

102.2 PARAMETERS

FACE GROUP

The group of faces to be projected onto the spline surfaces. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

SURFACE GROUP

The group of surfaces to project faces onto. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

CYCLE TYPE

all on each in surface in sequence

If projected parametrically, the spatial relationship between the faces will be preserved. For example, if the text “hello” is projected parametrically and in sequence, the letters will appear on the surface naturally, side by side. Sequence is not applicable if projecting along a vector.

all on each surface overlapping

If projected parametrically, the spatial relationship between the faces is not preserved, so they will appear overlapped in the area of the domain chosen for projection. Overlapping is not applicable if projecting along a vector.

one per surface

Each face is projected onto a surface in the order in which the face and the surface appear in the input or in their respective groups.

cycled

This is the same as *One Per Surface*, above, but starts the faces again if there are surfaces left.

ALONG VECTOR

The face is projected along a 3D vector and its image on the surface is converted into a curve on the surface. One spatial curve may generate several profiles depending on its position relative to the surface, the shape of the surface, and the chosen projection side. If the projected face does not intersect the surface at all, no profile curve will be generated.

axis

The axis along which the four corners of the feature are projected onto the base. When adding the feature from the outside as a Vector paste.

$X/Y/Z$

Cartesian axis - X, Y, or Z.

Face Normal Projection axis occurs along each face's normal. Thus, each face could potentially have a different projection direction than the others. The option also works well for non-planar faces.

Minimum Distance Projects the entire 3D face along the vectors of minimum distance to the surface. The projection resolution is given by the *Divisions per Span* value.

User Defined Enables the *Vector* fields immediately below to specify the X, Y, and Z components of the vector.

Note: When the *Minimum Distance* option is selected, the *Side* button becomes disabled because it's irrelevant for this projection type.

vector */projvec1, 2, 3*

The X, Y, and Z components of the projection vector if none of the main axes is chosen in the Axis parameter above.

side

Closest Will project the face onto the closest part of the surface in either direction of projection vector. For example, if a curve being projected onto a tube is inside the tube, it will yield two profiles, one on each side of the tube; if the curve is outside the tube, it will project onto the side closest to it.

Farthest The reverse of *Closest*. In the example above, *Farthest* would also yield two images if the curve is inside the tube, but it will choose the farthest side if the curve is outside the tube.

divisions per span */sdivs*

The number of points to be computed on the spatial face between successive spans. A span is the line connecting two consecutive CVs on a polygon, or the arc between two breakpoints on a spline curve. The projection tends to become more accurate as the number of divisions increases.

ray tolerance */rtolerance*

Controls the precision of the ray intersection with the surface. The ray is cast along the projection vector from every point of the 3-D curve.

fit tolerance */ftolerance*

Controls the 2-D fitting precision and is typically less than 0.01.

max uv gap (%) */uvgap*

This specifies what percentage of the size of the surface domain is acceptable for two separate profiles to be joined into a single curve.

order */order*

The spline order of the resulting profile curve. The type of curve (Bézier or NURBS) is inherited from the spatial curve. The order, however, is not inherited because the spline order provides useful control over the quality of the fit. If the spatial face is a polygon, the profile will be a NURBS curve.

preserve sharp corners

Controls the precision with which sharp corners in the projection curve are interpolated. It should be on when the projection has areas of high changes in curvature.

super accurate projection

Use a very accurate yet expensive algebraic pruning algorithm to determine the intersection of the vector with the surface.

PARAMETRICALLY

The spatial size of the faces is mapped directly onto the domain of the spline surface. The resulting projection is sensitive to the parameterization of the surface and is likely to appear distorted. It is, however, the fastest of the two projection types and tends to behave well on surfaces with regular shapes and chord-length parameterizations.

mapping type

Uniform mapping converts the spatial coordinates of the projection faces to (U,V) points in the domain of the surface without taking into account the parameterization of the surface. *Chord Length* mapping takes into account the surface parameterization and attempts to compute a projection best suited to the spatial and parametric determinants of the model.

u from

Specifies which of the spatial coordinates – X, Y, or Z – must be mapped to the U parametric direction of the surface.

v from

Specifies which of the spatial coordinates – X, Y, or Z – must be mapped to the V parametric direction of the surface.

map profile to range

This option is on by default. It causes the profile to be scaled and translated to fit within the surface's domain ranges described below. If this option is off, the profile's coordinates are mapped onto the surface domain without any transformation; consequently, the profile will not be visible if its points are not inside the domain of the target surface.

Typically, the projected face should not be mapped to the U/V range if it was previously extracted from the same surface using the Profile OP with the "Parametrically to XY" option selected (see *Parametrically to XY* p. 704).

The extraction and re-projection tandem can be very useful in achieving the modeling goals currently attainable only with 3D curves. Such tasks include the ability to edit the points of a profile, joining, stitching, or filleting profiles together, carving or refining profiles, etc. By extracting a profile parametrically to XY, Houdini creates a 3D face that is identical to the 2D profile but has an additional (constant) Z component.

The resulting 3D curve can be modeled using all the 3D tools available. Finally, the modeled 3D face can be reapplied onto the surface parametrically, making sure that the range mapping option is off.

u range */urange1, 2*

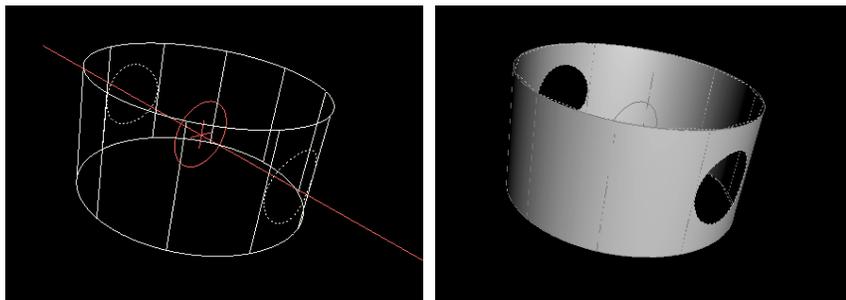
Indicates in percentages what part of the U surface domain is the mapping area. A full range of 0-1 will cause the profiles to be mapped to the entire domain in the U parametric direction. The range is not restricted to the 0-1 interval.

v range */vrange1, 2*

Indicates in percentages what part of the V surface domain is the mapping area. A full range of 0-1 will cause the profiles to be mapped to the entire domain in the V parametric direction. The range is not restricted to the 0-1 interval.

102.3 EXAMPLE

1. Place a Circle OP – Type: NURBS, Radius 0.3, 0.3; then place a Tube OP – Type: NURBS.
2. Connect them into a Project OP – Circle to Input1; Tube to Input2.
3. Append a Trim OP to the Project OP – it trims the surface according to the projection provided by the Project OP. You need to append a Trim OP to a Project OP in order to realise this trimming action.
4. Enable the template flags on the Circle and Tube OPs; make the Project OP the display OP.



102.4 TIPS

NURBS SCALP PATCH FOR HAIR

The: *Along Vector* sub-page > *Axis* > *Minimum Distance* option is extremely useful, say in the following situation: You have a NURBS surface of a head, and you want to obtain a NURBS patch by projecting a hairline onto the head's surface. You could:

- Template the NURBS head.
- Enable the “Snap to Template” option in the Model Editor (Snap options).
- Draw a NURBS curve along the surface of the head where you want the hairline.
- Then, with the Project OP, you select the *Minimum Distance* option, and you have a profile curve with which you can trim the surface of the head to obtain the patch.
- Thus having obtained a NURBS patch for the scalp, you could use it as the source, say for the Emit POP to grow a particle system in order to produce hair.

102.5 SEE ALSO

- *Bridge OP* p. 463
- *Delete OP* p. 543
- *Group OP* p. 592
- *Primitive OP* p. 697
- *Profile OP* p. 704
- *Trim OP* p. 819

I03 RAILS OP

I03.1 DESCRIPTION

The Rails OP generates surfaces by stretching cross-sections between two rails. This is similar to the Sweep OP, but it gives more control over the orientation and scaling of the cross-sections. The first OP input is the cross-section which will be replicated, and is typically placed in the XY plane. The second input OP source is the rails along which the cross-section is replicated.

I03.2 PARAMETERS

CROSS-SECTION GROUP

You can use a subset of primitives from the Cross-section input by specifying a group here. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

PATH GROUP

You can use a subset of primitives from the Rails input by specifying a group here. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

CYCLE TYPE

All Primitives at each point Places all the primitives from the Cross-section input at each point on the backbone.

One Primitive at a time Similar to the above, except the transformation is applied to individual primitives rather than to the whole.

Cycle Primitives This cycles through incoming primitives when placing them on a backbone. i.e. Start with 0 at vertex 0, primitive 1 at vertex 1, etc.

SWEEP ALONG PAIRS OF RAILS

Sweeps along rail 1 & 2, 3 & 4, 5 & 6 etc. instead of 1 & 2, 2 & 3, 3 & 4 etc.

SWEEP ALONG FIRST AND LAST RAIL

Connects the cross-section between the first and last rails.

STRETCH TO RAILS

Stretches the cross-section geometry to the rail geometry.

USE VERTEX

Specifies two vertices of the cross section polygon to be placed on rail1 and rail2 respectively. Very useful, for instance, to keep the first vertex on rail1 and the seventh vertex on rail2.

CONNECTION VERTICES */vertex1 /vertex2*

The vertices at which the cross-section is connected to the rails.

SCALE */scale*

Global scaling of the cross-sections.

ROLL */roll*

Non-cumulative rotation of the cross sections around the backbone. All cross sections get the same rotation.

FIX FLIPPING

Option to correct the flipping when no direction vector is used and the two rails happen to cross each other causing the normal to flip upside down.

USE DIRECTION

Uses the direction vector specified in the X, Y and Z coordinates. Otherwise it will use the normals of the geometry.

DIRECTION */dirx /diry /dirz*

The direction vector to use.

CREATE OUTPUT GROUPS

Selecting this option enables the creation of groups. A group is created for each backbone that is incoming. This allows for easy skinning in the Skin OP.

GROUP NAME

Specify the name of your output groups in this field. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

I03.3 INPUTS / GEOMETRY TYPES**INPUT 1**

The Cross-section geometry.

INPUT 2

The Rails geometry.

I03.4 SEE ALSO

- *Sweep OP* p. 791

I04 RAY OP

I04.1 DESCRIPTION

The Ray OP is used to project one surface onto another. Rays are projected from each point of the input geometry in the direction of its normal. This can be used to drape clothes over surfaces, shrink-wrap one object with another, and other similar effects.

I04.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

TRANSFORM POINTS

If selected, it will transform the input points as defined below. Leave this off when only interested in updating the source point attributes.

INTERSECT FARTHEST SURFACE

If selected, this option allows the user to choose between intersecting with the closest intersecting object or the furthest. * *See Example, below.*

POINT INTERSECTION NORMAL

If selected, updates each point in the source geometry with the normal at the *collision surface* it intersects with. If the point doesn't intersect at the collision surface, a normal of (0,0,0) is used.

POINT INTERSECTION DISTANCE

If selected, updates each point intersected with the distance to the collision surface. If the point doesn't intersect at the collision surface a distance of 0 is used. This value is placed in the \$DIST point attribute, accessible from the Point OP.

SCALE

/scale

A value of zero will leave the input point unaffected. A value of one will land the point on the intersection surface. Negative values and values > 1 are also valid.

LIFT */lift*

This value further offsets the surface input by offsetting it in the direction of its normal.

SAMPLE */sample*

This value determines the number of rays sent per point. If greater than one, the remaining rays are perturbed randomly, and averaged.

JITTER SCALE */jitter*

Controls the perturbation of the extra sample rays.

SEED */seed*

Allows a different random sequence at higher sampling rates.

CREATE POINT GROUP

If selected, it will create a point group containing all the points which were intersected successfully.

RAY HIT GROUP

Specifies the name of the above point group.

104.3 LOCAL VARIABLES

DIST Distance of the point to an intersected surface. This can be used in expressions by the Point OP.

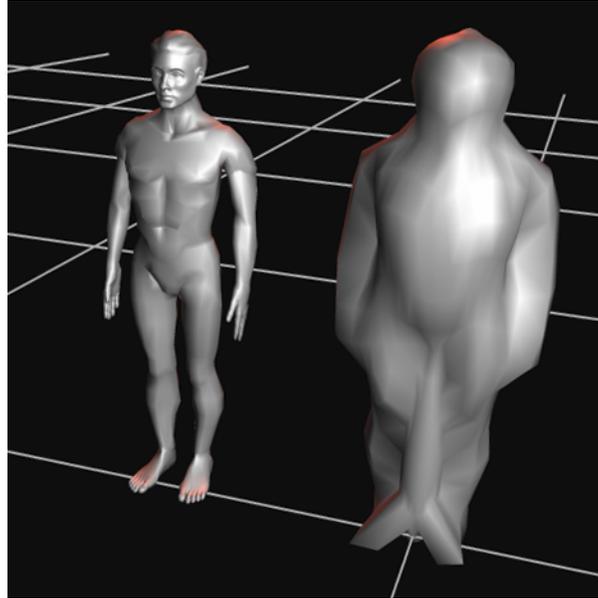
104.4 EXAMPLE

1. Place a Grid OP and translate it in Z by 2.5. Turn it's template flag on.
2. Append a Point OP to the Grid and enable the *Create Point Normals* option.
3. Place a NURBS Sphere with a *Radius* of 2,2,2 and translate it in Z by -2.5 .
4. Display point normals by enabling the option in the *Viewport > Display Options*.
5. Append a Ray OP to the Point OP and connect the Sphere to the right input. Make it the display OP.
6. Toggle the *Intersect Farthest Surface* button on and off.

The Ray OP will move the points of the Grid in the direction of the point normals. The first surface of the Collision Source (right input) will be where those points of the grid will rest. You can make those points rest on the other side of the sphere by

enabling the *Intersect Farthest Surface* option. This means that the points should continue to project to the farthest surface of the collision source.

104.5 SAMPLE OUTPUT



Sample output of the Ray OP.

I05 REFINE OP

I05.1 DESCRIPTION

The Refine OP allows you to increase the number of CVs in any NURBS, Bézier, or polygonal surface or face without changing its shape. It is also used to decrease the number of CVs within a given tolerance (i.e. a simple but fast method of data reduction).

THE DIFFERENCE BETWEEN REFINEMENT AND UNREFINEMENT

Refinement and unrefinement work both on faces (polygons, Bézier curves and NURBS curves) and surfaces (primitive meshes, Bézier surfaces and NURBS surfaces). To unrefine a face or a surface you need to specify a parametric interval (not just a single value as in refinement). This allows you to unrefine primitives within arbitrary intervals, either locally or globally. For example, to unrefine the whole primitive choose 0 and 1 as the two parametric boundaries; [0,0.5] will unrefine only the first parametric half of the primitive.

The interval boundaries are given by the First/Second U/V fields. Since refinement does not need an interval, the Second U/V fields are disabled by default.

The *Tolerance* control is only available for unrefinement, and not for refinement. Refinement does not need tolerances because it generates a curve or a surface that is mathematically identical to the original. Unrefinement, however, may tend to smooth out (or “melt”) the original in a given area. In short, unrefinement is lossy; refinement isn't.

I05.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

FIRST U / SECOND U /domainu1 /domainu2

This specifies a starting / ending location to complete the operation. Select this and a parametric U location between 0 and 1.

U DIVISIONS

If refining or sub-dividing, this option specifies the number of refines to be performed between *First U* and *Second U*.

FIRST V / SECOND V /domain1 /domain2

This specifies a starting / ending location to complete the operation. Select this and a parametric V location between 0 and 1.

V DIVISIONS

If refining or sub-dividing, this option specifies the number of refines to be performed between *First V* and *Second V*.

REFINE OPTIONS

If enabled, will refine the primitive at the locations specified above. If the primitive is a NURBS, then a knot is inserted multiple times as described below.

nurbs count u / v /refineu /refinev

Number of knots to insert at each location in the U / V basis when refining NURBS.

uniform domain lengths

Refine at equal basis intervals if refining a spline.

uniform arc lengths

Each refinement is done at the centre of the maximum knot span (splines) or edge length (polygons).

Note: Uniform Domain Lengths should be selected if the domain ranges are animating, otherwise the inserted points will be undeterminable as different maximum lengths are encountered.

UNREFINE OPTIONS

If enabled, will remove CVs from any NURBS curve or surface, polygon or mesh (points/ rows removed) between *First U* and *Second U*, and *First V* and *Second V*.

nurbs count u / v /unrefineu /unrefinev

Number of knots to remove at each location in the U / V basis if refining NURBS.

tolerance u / v /tolu /tolv

Only remove knots that do change the curve, polygon, or surface by more than this distance.

SUBDIVIDE OPTIONS

Subdivide refines a primitive such that the subdivision causes a sharp discontinuity if ever displaced. In essence subdivide is equivalent to refine for polygons and Béziers, since any refinement causes a potential discontinuity. In the case of a NURBS it is equivalent to a maximum refinement (i.e. count = primitive basis order - 1).

uniform domain lengths

Subdivide at equal basis intervals if refining a spline.

uniform arc lengths

Each subdivision is done at the centre of the maximum knot span (splines) or edge length (polygons).

Note: Uniform Domain Lengths should be selected if the domain ranges are animating, otherwise the inserted points will be undeterminable as different maximum lengths are encountered.

I05.3 INPUTS / GEOMETRY TYPES

Accepts any face type (polygon, NURBS curve, Bézier curve), or any hull type (mesh, NURBS surface, Bézier surface).

I05.4 LOCAL VARIABLES

none.

I05.5 SEE ALSO

- *Carve OP* p. 486
- *Fit OP* p. 577
- *Resample OP* p. 723

I06 RENDER PREVIEW OPERATION -

I06.1 DESCRIPTION

The Render Preview Operation allows you to render a portion of the Viewport as if it were a rendering. This is extremely useful for testing out lighting and materials, and getting an exact idea of how something will look in the finished scene.



In order to this, you select an output driver from which to render, and then draw a rectangle within the Viewport to define the area to render.

When a display driver has been chosen, Houdini begins rendering. When materials are changed, the display will be updated reflecting those changes (unless the lock button is turned on).

If there are changes other than to the material, choosing the output driver again will cause a new image to be rendered. Otherwise, only material changes are registered in the render. Changing lighting parameters will require another render.

The display mode will ignore any command-line options to *mantra* specified in the display driver, as well as output resolution and anti-aliasing information. However, the visible objects, dither and gamma correction information will be used.

I06.2 SUB-ICONS

PARAMETERS BUTTON

Displays the parameters dialog for this Operation. Empty, because there are none.

OUTPUT DRIVER MENU

Select an Output Driver with which to render the preview area.

I07 RESAMPLE OP

I07.1 DESCRIPTION

This OP will resample one or more primitives into even length segments.

The Resample OP only applies to polygons so when presented with a NURBS or Bézier curve input, it first converts it to a polygon using the *Level of Detail* parameter before continuing.

I07.2 PARAMETERS

GROUP

Allows you to specify which primitives or group of primitives to resample. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

LEVEL OF DETAIL

If the geometry you are performing this operation on is made up of something other than polygons (e.g. a NURBS circle), this parameter determines how detailed the conversion to polygons will be. The higher the level of detail, the more precise the resampling process.

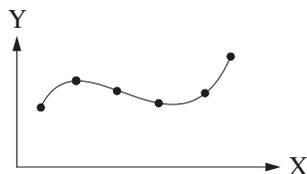
RESAMPLE BY POLYGON EDGE

This toggle allows the resampling to be done on a per-edge basis of the polygon rather than on the polygon as a whole. This means that corners will be preserved in the resampling. When this parameter is enabled, the measure method will be disabled. If the segment length is specified, only edges which are longer than the specified length will be subdivided. Each edge will be evenly subdivided so that the maximum length of each subdivision is less than or equal to the length specified.

If the maximum segments is chosen, the edges will not be split into more segments than are specified. If the length parameter is turned off, then each edge will be split into the specified number of segments.

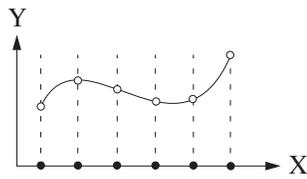
METHOD

even length segments



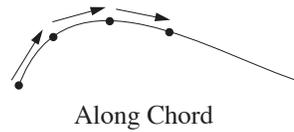
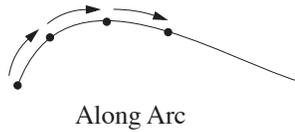
Even-length measures are made along the original input.

even x / y / z segments



Even-length measures are made along the X/Y/Z axes and are then superimposed onto the original input. The spacing may not necessarily be even along the original input itself using these three methods.

MEASURE



along arc

This method of measurement divides the curve itself into even length segments.

along chord

This method creates straight even-length segments between the evaluated points on the primitive. This is faster but not as accurate.

MAXIMUM SEGMENT LENGTH

When this option is enabled, it overrides the value of the Arc/Chord length and allows you to specify your own values for the length of the segments in the resampled geometry. Smaller values produce smoother interpolation.

LENGTH

This parameter lets you specify the Arc/Chord length for the resampled geometry.

MAXIMUM SEGMENTS

When enabled, allows you to override the maximum number of segments used in the resampled geometry.

SEGMENTS

Lets you specify the number of segments used when measuring along arc.

When both *Maximum Segment Length* and *Maximum Segments* are selected, the resultant polygon is constrained to a fixed length which may be shorter than the original input curve.

When only *Maximum Segments* is selected, the segments span the full distance of the original input curve.

When neither option is selected, the input curve is merely converted to a polygon.

MAINTAIN LAST VERTEX

If selected, the primitive's final CV is included in the resampled polygon, even if the final resulting edge is shorter than the given segment length.

I07.3 INPUTS / GEOMETRY TYPES

The Resample OP works in conjunction with any primitive type that can be converted to polygons. This includes Bézier curves, NURBS curves, circles, and polygons.

I07.4 USES / WORKS IN RELATION WITH

- *Convert OP* p. 512
- *Refine OP* p. 719

I07.5 EXAMPLE

To create a constant length flag with the Resample OP:

1. Create a number of horizontal lines using a Grid OP:
Primitive Type: Polygon; Connectivity: Rows; Size: 3, 1
2. Append a Group OP: Group: group1; Entity: Points;
Number: Enable; Pattern: 0-99:1,10
3. Append a Spring OP: Source Group: group1; Forces/Wind: 1, 0, 0;
Turbulence: 1, 0, -1
4. Clip each line to a constant length by appending a Resample OP:
Maximum Segments: On; Segments: 15
5. Appending a Skin OP whose parameters can stay at the default, and enable the Points display in the Viewport options.
6. Click *Play* to view the result. Adjust the view as necessary.
7. Toggle the display OP back and forth between the Spring OP and the Skin OP. Notice how the lines are stretched and deformed from their original length by the forces acting on them in the Spring OP. After resampling, they are a constant length.

I08 REST POSITION OP

I08.1 DESCRIPTION

This OP is used to create an attribute which causes material textures to stick on surfaces deformed using OPs (i.e. Magnet, Transform, Skeleton). It takes non-deforming geometry which matches the topology of the deforming geometry. It then sets an “attribute” to store the static geometry. Once this attribute is set, shading will happen using the “rest position”, instead of the deformed position.

The OP can get the rest position in one of two ways:

- By reading a file (defined in the *Assign From File* parameter).
- By attaching a second input.

The first input is the deforming geometry. The second input is the rest position data, which should be a static, non-deforming surface. The topologies of the two geometries should match.

All primitives support the *rest* attribute, but in the case of quadric primitives (circle, tube, sphere, and most particle renderers), the rest position is only translational. In other words, it will not reflect the rotational or scale transformations of the primitive.

The rest attribute is exported to RenderMan as the *Pr* attribute. This is output for all surfaces as a “Vertex Float Attribute”.

I08.2 PARAMETERS

ASSIGN FROM FILE

Enter a file pathname here for the rest position geometry, if you aren’t inputting the rest position via the second input.

REST NORMALS

These normals are used in feathering to ensure that feathering occurs on the rest geometry (as opposed to the deformed geometry). Most primitive types are handled internally by *mantra* (i.e. rest normals are not required for NURBS). However, when applying feathering on polygons, rest normals are required.

If the rest normal does not exist, the polygon normal (computed from the rest position) will be used by *mantra* – resulting in a “faceted” look to the feathering.

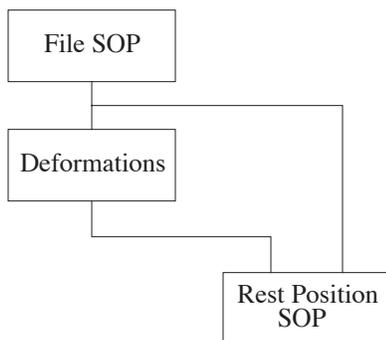
108.3 DISCUSSION

When OPs modify geometry, the shapes often deform. In order to get a texture to “stick” to the deforming surface, there are two approaches. The first is to attach texture coordinates to the geometry using a Texture SOP *before* the deformation occurs. The second is to use a “rest position”.

A rest position is a static (i.e. non-deforming) piece of geometry which is used for shading purposes only. By using a Rest position SOP, a geometry attribute called *rest* is attached to the geometry. This attribute is used by the Material Editor to compute the texture positions.

The Rest position SOP takes non-deforming geometry which matches the topology of the deforming geometry. It then sets an “attribute” to store the static geometry. Once this attribute is set, shading will happen using the “rest position”, instead of the deformed position.

108.4 EXAMPLE



Two examples in which we might want to use rest positions: i) if we route an object through a Twist SOP for a bend; and ii) if we model a leaf, apply a texture, and copy it to the points in a Particle SOP. How do we get the textures to stick?

From the illustration above, we can see the general case for the use of the Rest Position SOP. For the Twist SOP, the deformations box would simply be the Twist SOP.

For the leaves deforming, you simply feed the source into both inputs of the Rest Position. This assigns the shading information for the leaf. Then you can copy this leaf to a particle system and have the shading occur based on the original geometry.

PROCEDURE

To see this yourself, try the following:

1. Create a Grid OP, feed it into a Twist OP and animate the twist.
2. Apply a material with an orthographic projection to the grid.
3. When the grid is being shaded, the orthographic projection in the material is being done on the translated geometry, so you will see that the object “swims” through the texture. In most cases, this is not desirable.

If you take the output of the twist and feed it into a Rest position OP, then take the original (non-deforming) grid and feed it into the second input of the Rest position OP, a rest position attribute will be created. Now, the texture should seem to stick to the object. In fact, the material is now using the position of points on the original grid when calculating the texture coordinates. It's as if the original grid was being shaded, then, the deformation occurs.

108.5 REST ATTRIBUTES FOR RENDERMAN

In Houdini, the rest position attribute is named *rest*. By default, this attribute will be passed down to RenderMan as a “vertex point” attribute named “Pr”. If you are writing your own shaders, it is possible to access the rest position instead of the object position for shading purposes. To do this, you must declare the “Pr” attribute as a parameter to your shader. Then, simply use the “Pr” attribute where you previously would have used the “P” attribute.

For example:

```
surface
myShader(float myparm=1; point Pr=0)
{
    point    PP;
    float    amp;

    PP = transform("shader", Pr);
    amp = noise(PP) * myparm;
    ...
}
```

Note: The Pr attribute is declared after all of the user defined parameters for the shader.

Of course, it is possible to build your own attributes and pass them down to shaders in this fashion. Or it's also possible to re-define the default Houdini attribute mappings by using the Attribute SOP.

108.6 INPUTS

INPUT 1 – DEFORMING (GEOMETRY)

This input takes in the geometry that is being deformed.

INPUT 2 – REST POSITION (GEOMETRY)

This input provides the OP with a copy of the geometry in it's “rest position”; i.e. what it's like before being deformed.

Note: The topologies of the geometry in both inputs should match.

I09 REVERSE OP

I09.1 DESCRIPTION

The Reverse Operation allows you to reverse or cycle the vertex order for all faces. It provides functionality from from Face/Hull folder in *Primitive OP* p. 697.

I09.2 PARAMETERS

SOURCE GROUP

Primitive and/or profile group to operate on.

VERTEX

<i>Reverse</i>	Reverses U for faces, U & V for hulls.
<i>Reverse U/V</i>	Reverses U or V.
<i>Swap</i>	Interchanges U, V. Preserves topology.
<i>Shift</i>	Cycles vertices by U/V Offset.

U OFFSET */vtxuoff*

Amount to cycle vertices in U direction.

V OFFSET */vtxvoff*

Amount to cycle vertices in V direction.

I 10 REVOLVE OP

I 10.1 DESCRIPTION

This OP revolves faces to create a surface of revolution. The revolution's direction and origin are represented by guide geometry that resembles a thick line with a cross hair at the centre. The cross hair represents the origin of the revolve as entered in the dialog and the stick represents the direction. Changing any of these parameters will cause the guide to change appropriately.

If the guide geometry is too distracting, you can disable it by entering the Viewport options dialog and clicking on the Guide geometry button so that it no longer appears indented. This procedure is global and will disable the guide geometry of other OPs as well.

I 10.2 PARAMETERS – REVOLVE PAGE

CONNECTIVITY

This option is used to select the type of surface, when using a *Mesh Primitive Type*.

- Rows Creates horizontal lines.
- Columns Creates vertical lines.
- Rows & Cols Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon).
- Triangles Build the grid with Triangles.
- Quadrilaterals Generates sides composed of quadrilaterals (default).
- Alternating Triangles Generates triangles that are opposed; similar to the Triangles option.

ORIGIN

/originx /originy /originz

Coordinates defining the origin of the revolution.

DIRECTION

/dirx /diry /dirz

X, Y, and Z coordinates of the direction vector defining the direction of the revolve.

CONVERT MESH TO POLYGONS

Changes the output mesh to consist of individual polygons.

IMPERFECT

Applies to splines only. If selected, the results are approximated nonrational curves, otherwise they are perfect rational curves.

110.3 PARAMETERS – DETAIL PAGE

REVOLVE TYPE

Determines how the revolve should be generated.

<i>Closed</i>	An enclosed curve.
<i>Open Arc</i>	An open curve segment.
<i>Closed Arc</i>	An open arc with connecting edges to the centre resembling a slice of pie.

START / END ANGLES */beginangle /endangle*

The start and end angles of the revolve. A revolve will start at the beginning angle, and proceed towards the ending angle. If beginning=0 and end=360 it will be fully revolved. Values greater than 360° are also valid.

DIVISIONS

Density of the resulting mesh surface.

ORDER

If a spline type is selected, it is built at this order.

END CAPS

If selected, it adds faceted end caps to the ends of the revolved surface. See the *Cap OP* p. 468 for a full description of caps.

110.4 INPUTS / GEOMETRY TYPES

Polygon, Bézier curve, NURBS curve.

110.5 LOCAL VARIABLES

none.

110.6 USES / WORKS IN RELATION WITH

Creating surfaces of revolution such as glasses, candlesticks, columns and other objects requiring a revolved profile.



III ROUND OBJECT

III.1 DESCRIPTION

This OP generates round fillets of a specified radius between two surfaces. It can also be used to round closed surfaces, as it will generate spherical patches to fill in the corners.

Each surface in group 1 is compared with each surface in group 2. If they come within the specified radius of each other, a rounded fillet is generated between them. The direction of the fillet is determined by the normal of the surfaces, modified by the *Flip Fillet Direction* parameters. For best results, ensure that the normals of all surfaces face outward from the centre of the object being rounded. Note that self-fillets are not performed, even if a surface is in both group 1 and 2.

III.2 PARAMETERS

ROUND GROUP 1 / 2

Subsets of NURBS and Bezier surfaces.

3D TOLERANCE */tol3d*

World space precision of fillet.

2D TOLERANCE */tol2d*

Domain space precision of fillet.

MARCHING STEPS */step*

Number of steps for tracing each profile span.

RADIUS */radius*

Radius of the fillet.

SPAN DENSITY */spans*

Density to build fillet. Higher densities will provide a more accurate rounded fillet at the expense of more geometry. The actual number of isoparms in the fillet is a function of this, the radius, and the length of the fillet.

FLIP LEFT/RIGHT FILLET DIRECTION */flipnorma /flipnormb*

Logically flip normals of left/right surfaces.

TRIM PRIMITIVES */trimedges*

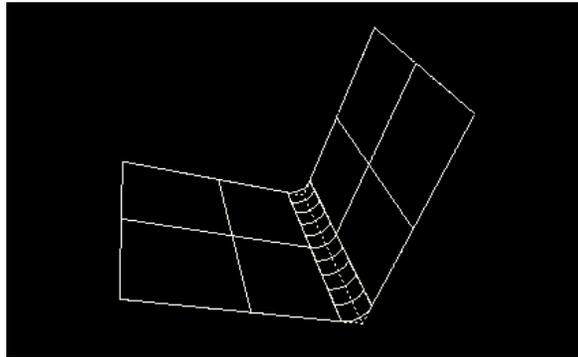
Try to cut surfaces back to the fillet.

CUSP ANGLE */angle*

Angle at which a sharp corner is made.

III.3 EXAMPLE

1. Place a Grid OP and make it a NURBS type.
2. Append a Copy OP set to make 2 copies with a Translation of: 0, -0.5, 0.5 and Rotation of: -90, 0, 0 . The result is two planes placed at right angles.
3. Append a Round OP to the copies, and set the display flag. Notice that a curve is drawn between the planes and part of the grids have been trimmed. The direction of the rounding is determined by the base surface normals so the result should be a rounding of the join between the planes with the default settings.



4. Try adjusting the Radius, Span Density, and other parameters. If the round doesn't work, first check the surface normals; they should be oriented the same.

III.4 SEE ALSO

- *Bridge OP* p. 463
- *Fillet OP* p. 574
- *Surfsect OP* p. 788

I 12 RMAN SHADER OP

I 12.1 DESCRIPTION

The RMan Shader OP allows you to attach different RenderMan shaders to different groups of primitives – as opposed to a single shader specified at the *Object level > Shading* page. Note that the shader specified at the Object level serves as the Default shader if there are primitives that aren't affected by a RMan Shader OP.

The groups used in this OP can either be specified by numeric indices to the primitives, or by the names of groups defined in prior OPs (like from a).

There is space to assign up to six different RenderMan shaders to six (possibly different) groups of primitives. If that is inadequate, then you can string a sequence of RMan Shader OPs together to assign more. Note that it is possible to override a shader assignment when doing this.

I 12.2 PARAMETERS

GROUP(S)

Specify the groups in question. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

SURFACE SHADER(S)

Name of RenderMan Shader to assign to the specified group(s).

renderman shader dialog

Clicking on the  button beside the *Surface Shader* field provides a short-cut to defining parameters that are passed to the RenderMan shader. Select a RenderMan shader from the pop-up menu, check off the options you desire, and the parameter string will be built for you, and inserted into the *Surface Shader* field.

I 12.3 SEE ALSO

- *Group OP* p. 592
- *Material OP* p. 634

I 13 SCULPT OP

I 13.1 DESCRIPTION

The Sculpt brush is designed to allow you to adjust the shape of geometry using a brush. The deform mode will displace the geometry based upon the current displacement amounts. The smooth mode will just smooth out the geometry.

This operation is meant to be used interactively in the viewport.

I 13.2 PARAMETERS – OPERATION PAGE

GOAL

Deform Points	Moves each point along its normal.
Smooth Points	Moves each point towards the average of its neighbours. This is a per dab operation, thus does not accumulate a stencil buffer.
Deform and Smooth	Applies both the deformation and a smoothing pass.

AXIS, VECTOR */vector*

Determine what direction the deformation occurs in. "Normal" refers to the surface normal at the point; the others are in the object's coordinates.

ACCUMULATE TO STENCIL

By default, the operation is applied and the stencil cleared after every brush stroke. With this set, the stencil is not cleared until "Apply and Clear Stencil" is pressed.

APPLY & CLEAR STENCIL

Applies the operation and clears the stencil. This is equivalent to right-clicking in the viewport.

UPDATE DISPLACEMENT NORMALS

Recomputes the point normals used by the sculpting operation.

LMB / MMB DISPLACEMENTS */fg/bg*

Specify the amount by which to displace the surface.

If "Accumulate To Stencil" is on, the left mouse (LMB) does foreground and the middle mouse (MMB) erases.

APPLY TO ALL

Deforms all of the selected geometry.

RESET ALL CHANGES

Restores geometry to initial state.

113.3 PARAMETERS – BRUSH PAGE

SHAPE

The basic shape of the brush: circle, square, or bitmap.

BITMAP

What bitmap to use. The alpha channel becomes the brush.

RADIUS */radius*

The radius of the brush.

BRUSH ANGLE */brushangle*

How far to rotate the brush.

BRUSH SQUASH */squash*

Amount to squash the brush in the y direction before rotation.

OPACITY */opacity*

The amount to affect the stencil mask.

BRUSH SPLATTER */brushsplatter*

A random noise in the brush's opacity based upon the position on the brush.

PAPER GRAIN */papergrain*

A random noise on the object's stencil mask based on the object position.

SOFT EDGE */softedge*

Percentage of the brush to be rolled off.

KERNEL FUNCTION

Which metaball kernel to use for the roll off.

UP VECTOR TYPE

How the brush should be oriented on the surface:

Stroke Direction	Oriented in the direction in which the brush moves.
Fixed	Oriented as specified in the Up Vector field.

UP VECTOR */upvector*

The fixed up vector to orient brush to.

113.4 PARAMETERS – STROKE PAGE

Note: Be sure to see: Interface > Secrets of the Brush Page p. 142 for useful info!

ORIENT BRUSH TO SURFACE

Switches between the brush being perpendicular to the surface or always oriented along the view direction. If you are having trouble with a shaky brush, try turning this off.

REALTIME MODE

Enable this to paint geometry as it's deforming.

The last few parameters on this page are read-only parameters. While it is theoretically possible to manually apply brush strokes by adjusting these parameters, that would be excruciatingly difficult. Their primary use is to provide feedback as to where the brush is currently hitting your geometry. This can be used as a sort of 'geometry eyedropper' when you want to quickly investigate some troublesome geometry.

DIRECTION

This is the normal on the hit primitive at the hit position.

HIT LOCATION

This is the position in space where the brush ray hit the geometry.

HIT PRIMITIVE

This is the primitive number that was hit by the brush ray.

HIT UV

These are the parameteric coordinates where the primitive was hit. They are the UV coordinates used to evaluate the surface with a prim call, for example, and is not to be confused with any "uv" texture coordinates that may be present.

EVENT

This is used internally to track the current state of the drawing mode. No-op is the default and is used to prevent any accidental writes using out of date or irrelevant hit information.

113.5 SEE ALSO

- *Capture Paint OP* p. 480
- *Comb OP* p. 504
- *Paint OP* p. 649

I 14 SELECT OP -

I 14.1 DESCRIPTION

The select operation lets you preselect geometry for the next operation.

The normal way to model in Houdini is to first choose the operation you would like to perform and then select the geometry on which to perform that operation (verb - noun). The select state allows you to preselect some geometry that will be used for the next operation to perform.

I 14.2 GENERAL NOTES

Note: These apply to both the Select OP and the selection stage of other Operations.

SELECTION TYPES

The Selection Icons (see *Selection Icons* - p. 809) on the left of the Viewport let you change the type of selection (e.g. points, primitives, edges, etc.) The  -  keys can also be used to change the selection type.

SELECTION MODE

The default selection mode is "replace" mode, where newly selected geometry replaces the current selection. To quickly use "toggle" mode while in "replace", hold the shift key while selecting. While in "toggle" mode, clicking on already selected geometry will remove it from the selection, and clicking on unselected geometry will add it to the selection. The selection mode can be changed to "replace", "toggle", "add", or "remove" by clicking on the menu at the left hand side of the viewer.

BOX / LASSO / PAINT PICKING

The default selection style is "box picking". This means that clicking and dragging the mouse will draw a box, and the current selection rule will be applied to everything in the box. The selection style can be changed to "box picking", "lasso picking", or "paint picking" via the menu to the left of the viewer. Lasso picking lets you draw freehand selection regions. Paint picking lets you drag the mouse over geometry to select or unselect it. While paint picking, the left-mouse-button uses "replace" mode, Shift-left-mouse-button uses "add" mode, and the middle-mouse-button uses "remove" mode.

SELECT WHOLE GEOMETRY

The "select whole geometry" toggle at the left of the viewer lets you pick an entire geometry object by clicking on any part of it. The keyboard shortcut for this toggle is .

FRONT / BACK POLYGON MENU

The "front/back polygon" menu at the left of the viewer lets you select front and back facing polygons, only front facing polygons, or only back facing polygons. This menu is useful when working in the UV viewer.

NON-DRAGABLE SELECTING

Some operations, like transform and edit, let you select geometry and drag it while in selection mode. The moment you drag the geometry Houdini exits selection mode and begins to change the parameters of the current operation in order to drag the geometry. If you do not want to accidentally drag geometry, use the middle-mouse-button (**M**) to select it instead of the left-mouse-button.

114.3 SELECT OPERATION MENU (**Ctrl** **M**)

SELECT ALL **A**

Selects all the object / scene geometry (depending on the status of Select From All Objects).

TOGGLE SELECTION **T**

Unselects that which is selected, and selects that which was formerly *not* selected.

SELECT NONE **N**

Deselects everything.

SELECT FROM ALL OBJECTS

Normally, you work on the contents of one object within your scene at a time. Enabling this allows you to select from all the objects in your scene. Picking geometry from other objects will create object merges.

This is tied to the Select One/All icon at the bottom of the Viewport.

I 15 SEQUENCE BLEND OP

I 15.1 DESCRIPTION

This OP allows you do 3D Metamorphosis between shapes and Interpolate point position, colors, point normals, and texture coordinates between shapes.

I 15.2 PARAMETERS

BLEND FACTOR */blend*

This value determines the blend transition between consecutive geometry inputs. Values between 0 and 1 will control the metamorphosis between geometry input 1 and 2, values between 1 and 2 will control the metamorphosis between geometry input 2 and 3, and so on.

BLEND POSITION

If checked, only point XYZ positions are blended.

BLEND COLOR

Point Colors are blended.

BLEND NORMALS

Point normals are blended.

BLEND TEXTURE

Point texture coordinates are blended.

I 15.3 INPUTS / GEOMETRY TYPES

You can specify up to 9999 consecutive OPs (typically File OPs). It is extremely important that the number of Points and Polygons in each OP are *exactly* the same (check the Info of each source OP to ensure it is so). If you need to Cusp a blended object, do it after the Blend, as it may produce a different number of primitives for input.

I 15.4 USES / WORKS IN RELATION WITH

Generate geometry morphs between objects. The number of points between the inputs should be the same for best results but the point count between the input geometries can be different. For best results, the point or cv numbering of the input geometries should be the same in terms of position on the geometry.

Sequence blend differs from the Blend OP in that the former always creates a geometry blended from two inputs, while Sequence blend can blend an arbitrary number of inputs simultaneously.

I 15.5 SEE ALSO

- *Blend OP* p. 453

116 SHADER OP

116.1 DESCRIPTION

This OP will add attributes to the geometry to define a shader per primitive. The shader string is generated by evaluating a SHOP specified. Both surface and displacement shader strings can be generated by the Shader OP.

116.2 PARAMETERS – SURFACE PAGE

Allows you to assign a surface shader (SHOP) to geometry.

RENDERER

Specifies for which renderer you are assigning the shader.

GROUP

Specify a group within the geometry to which you want to apply the Shader. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

SHADER STRING

Pick a SHOP surface shader to apply. Click on the \oplus button to pick a SHOP shader file.

116.3 PARAMETERS – DISPLACEMENT PAGE

Allows you to assign a displacement shader (SHOP) to geometry.

RENDERER

Specifies for which renderer you are assigning the shader.

GROUP

Specify a group within the geometry to which you want to apply the Shader. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

SHADER STRING

Pick a SHOP displacement shader to apply. Click on the \oplus button to pick a SHOP shader file.

116.4 SEE ALSO

Scripting > VEX Section on how to make SHOP shaders.

I 17 SKELETON OP

I 17.1 DESCRIPTION

The Skeleton OP is used to animate characters that have been modeled statically. In order to deform geometry, its points must be caught within a hierarchy of circles (the skeleton). Each pair of circles defines a capture region which contains a subset of the input geometry points. As the circles are animated, the input source points are transformed accordingly.

I 17.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

SKELETON OBJECT

This parameter specifies the head of the primitive ellipse hierarchy defining the skeleton. The Skeleton OP extracts all ellipses from this object as well as all its children, building an internal skeleton of circles all transformed to a common space.

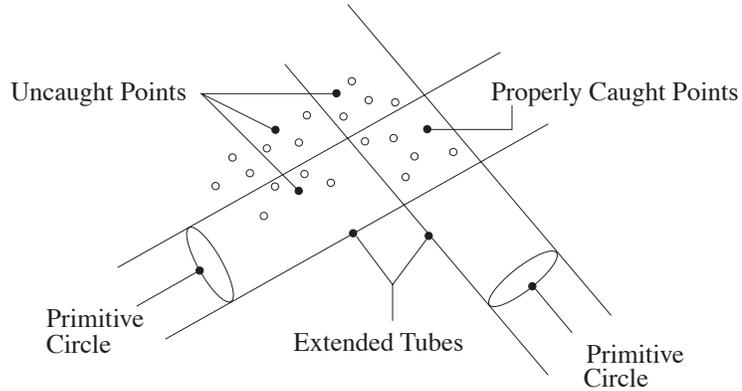
SOURCE OF GEOMETRY

You can tell the Skeleton OP where it should look in object hierarchies to find the ellipses that will be used to deform geometry. You can specify the render OP, the display OP, or a OP named “capture_sop” as the target. Bone objects contain OPs named “capture_sop”, and it is mostly for dealing with bone objects that this functionality has been added.

When doing a capture, the Skeleton OP will save the position and orientation of any bones in the hierarchy into their “capture angles” parameters for later use if the capture regions need adjustment.

CAPTURE POINTS / CLEAR CAPTURE REGIONS

These two function buttons are used to safely discard / recalculate the point assignments to each capture region if it lies within the intersection of both “tubes” defined by the extension of the capture region as illustrated below:



CAPTURE ALL POINTS

This option ensures that all points in the source geometry (that which will be deformed) are captured by assigning points to the closest volumes.

COLOR ATTRIBUTES

Point coloring can be used to derive which points have been caught, or will be caught at the next capture:

- Default Source Color* The point color attributes are not changed.
- Color by Capture Score* The color of the point is dependent upon the score as follows:
 - Red: not caught by any capture regions.
 - Yellow: caught by only one tube of a capture region.
 - Green: caught by both tubes of a capture region.
 - Blue: point is over-ridden to another capture region.
- Color by Capture Region* Each capture region is assigned a random color, and all points within that region are assigned that some color.

TRANSFER SELECTION TO CAPTURE REGION

This will transfer the current point selection to the capture region below.

CAPTURE REGION

This is used to select a single region for display or as a destination for transferring points to. If set to -1 (default), all regions are displayed and transferred points are uncaught (see Overriding Points).

CREATE OUTPUT GROUPS

If selected, a point group is created for each capture region containing the points within it.

GROUP PREFIX

If creating output groups, each is prefixed with this label, and ends with the capture region number.

GUIDE OUTPUT

Used to select between simple circles and tubular skeleton guide geometry. Note the tube output is not exactly the capture region defined by two circles, but simply an approximation. The actual region is the intersection of the tubes formed by extending both circles as described above.

117.3 OVERRIDING POINTS

The easiest way to override points is by selecting the points in question, displaying the desired destination capture region and selecting the transfer function. This is done by:

- Set *Color Attributes* to *Color by Capture Score*.
- Use the slider in *Capture Region* to highlight the desired capture region. If the slider is set to -1, the point selection is uncaptured.
- In the viewport, enter point selection mode, and box-select or individually select the bad points. They should become high-lighted as if they were selected in the Model Editor.
- Select *Transfer Selection to Capture Region*. The points will become blue (overridden), and moved to the capture region specified or, if the region is set to -1, the points become red.

117.4 TIPS

- It is very useful to capture all points for a successful skeleton. Use the score coloring to quickly debug the skeleton or add extra circles to capture unused areas.
- Use the information pop-up on the OP to display the number of capture regions found and the proportion of points safely caught.
- Only primitive circles can be used to define the skeleton. These show up as guide geometry.
- If you are used to working with polygon circles in *action*, and still desire this type of functionality, you can convert them to primitive circles (necessary for the Skeleton, Arm, and Limb OPs) using the Convert OP.

- If all else fails, the skeleton capture information is saved within the Houdini CPIO file, which can be expanded with the CPIO_Expand command (see the *Editing hip Scripts* p. 195 in the *Scripting* section for details) and edited separately. Its file extension is *.skel* or *.bskel*, and it contains comments describing the capture assignment format.
- Use “Default Source Colors”. This is because point coloring takes additional CPU time, and colors between points are smoothed out. If you want quick feedback on the basic shape of your deformed character, you can save a great deal of time by taking out any point colors you may have.

118 SKIN OP

118.1 DESCRIPTION

The Skin OP takes any number of faces and builds a skin surface over them. If given two or more surfaces, however, the OP builds four skins, one for each set of boundary curves.

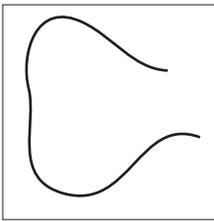
All face and surface types are valid as long as the input(s) contain only faces or only surfaces. Different face types can be skinned together into one surface. For example, it is possible to skin a cubic open NURBS curve with a polygon and a quintic closed Bézier curve even if the three faces have a different number of control vertices. Similarly, this OP can skin the boundary curves of surfaces of different types, number of rows, columns, etc.

When face types are input, the number of input OPs and the number of faces in each input establish the skinning method. If only one input exists, a “linear-skinning” operation is performed by running a skin across the cross-sections. The result is the classic ruled or skinned surface. If a second input exists, a “bi-linear skinning” is performed which computes a cross-skin between the faces in the first input (U cross-sections) and the faces in the second input (V cross-sections). The result is a surface whose name derives from the number of cross-sections in each direction: triangular, square, or multiple boundary surface, as well as a special case of swept surfaces and N-rails. When possible, cross-sections are interpolated as ioparms.

If you need more control over tangency in the skin, try using the *Bridge OP* p. 463.

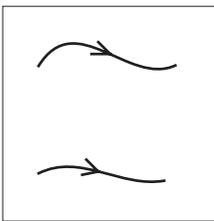
Tip: If you have problems with the results being skinned in the wrong order, try inserting a Sort OP ahead of the Skin OP, and *Sort by Normals*.

118.2 TYPES OF SURFACES



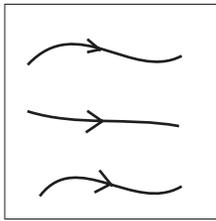
SINGLE BOUNDARY SURFACE

One face, open or closed, is converted into a surface whose boundaries match the shape of the face exactly. Basically, this operation builds an interior area for the face. The surface type will be similar to the type of the face. For example, a NURBS curve yields a NURBS surface. If the curve is highly concave, the result may look less satisfactory than expected.



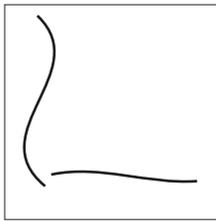
PATCH

Two boundary faces define a ruled surface. The arrows on the two faces indicate the required parametric direction, which must be the same for both faces to avoid a bad twist in the surface. Use the Primitive OP or the modeler to correct the problem. The surface type will be similar to the most complex type between the two cross-sections. For example, if a polygon and a NURBS curve are skinned together, the surface type will be NURBS. The surface always contains the two faces as two of its boundaries.



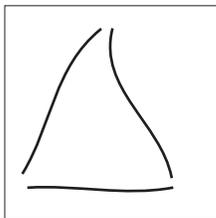
LINEAR RULED/SKINNED SURFACE

Two or more faces are skinned linearly into a single surface. The arrows on the faces indicate the required parametric direction of each face, which must be the same for all faces to avoid bad twists or flips in the surface. Use the Primitive OP or the modeler to correct the problem. The surface type will be similar to the most complex type among all cross-sections. For example, if a polygon, a Bézier and a NURBS curve are skinned together, the surface type will be NURBS. The surface goes through each cross-section unless “Preserve Shape” if OFF (see parameters below). If the cross-sections have repeated points, or share points between them, the result might not look good when shape preservation is enabled.



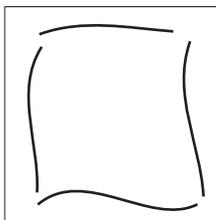
A SPECIAL SWEEPED SURFACE

This case does a bilinear skin and requires two inputs. The U face (1st input) is swept along the V face (second input). The two faces do not need to touch at their endpoints. If their endpoints coincide, though, the two of the surface’s boundaries will match the two faces exactly. The surface type will be similar to the most complex type of the two faces. For example, if a polygon and a Bézier curve are skinned together, the surface type will be Bézier.



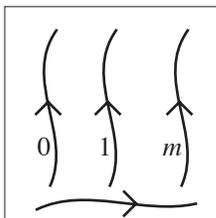
TRIANGULAR SURFACE

This case requires two inputs for the bilinear skin. One input has two faces; the other input, just one. The endpoints of the faces need not coincide, but if they do, the surface boundaries will match the face shapes exactly. Basically, the three faces define an interior area to be filled by a surface. The surface type will be similar to the most complex type among the three boundary faces. For example, if the faces are Bézier and NURBS curves, the surface will be a NURBS primitive.



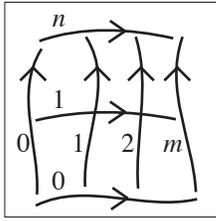
SQUARE SURFACE

Four faces define the outer boundaries of a surface. This case requires two inputs for the bilinear skin: the two U boundaries (1st input) are cross-skinned with the V boundaries (the 2nd input). The endpoints of the faces need not coincide, but if they do, the surface boundaries will match the face shapes exactly. Basically, the four faces define an interior area to be filled by a surface. The surface type will be similar to the most complex type among the four boundary faces. For example, if the faces are polygons and NURBS curves, the surface will be a NURBS primitive.



A SPECIAL CASE OF M-RAILS

One input contains the rails, and the other input the cross-section. The cross-section is swept along the rails to form a surface. The arrows on the faces indicate the required parametric direction of each face, which must be the same for all faces to avoid bad twists or flips in the surface. Use the Primitive OP or the modeler to correct the problem. The surface type will be similar to the most complex type among both rails and cross-section. For example, if the faces are polygons and NURBS curves, the surface will be a NURBS primitive.



MULTIPLE-BOUNDARY SURFACE

Not to be confused with N-ary patches. This case generalizes the square surface concept by allowing more interior cross-sections both in U and V. If no interior cross-sections exist, this case reduces to a square surface. The surface interpolates all the boundaries and the interior cross-sections. The result improves when the faces intersect. The arrows on the faces indicate the required parametric direction of each face, which must be the same for all faces to avoid bad twists or flips in the surface. Use the Primitive OP or the modeler to correct the problem. The surface type will be similar to the most complex type among all faces. For example, if the faces are polygons and NURBS curves, the surface will be a NURBS primitive.

118.3 PARAMETERS

U CROSS-SECTIONS

Empty by default, this field provides a way to specify a subset of the first input's faces and surfaces. Do so by selecting one or more primitive groups from this field's pop-up menu.

V CROSS-SECTIONS

Empty by default, this field provides a way to specify a subset of V faces. If a second input exists, the primitive groups available for selection are taken from the second input. If only the first input is present, the groups listed in the pop-up menu belong to the first input. This means that two inputs are not always needed for a bilinear skin as long as the first input has both U and V groups in it.

CONNECTIVITY

(Results only *viewable* for polygons and meshes).

- Rows Creates horizontal lines.
- Columns Creates vertical lines.
- Rows & Cols Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon). Compare them in the Model Editor.
- Triangles Build the grid with Triangles.
- Quadrilaterals Generates sides composed of quadrilaterals (default).
- Alternating Triangles Generates triangles that are opposed; similar to the Triangles option.

PRESERVE SHAPE

This parameter determines the precision of a linear skin (case c in the diagram). If enabled, it ensures that the generated surface goes through each cross-section. Here, a cross-section can be a face or a surface boundary, depending on the types being skinned. If disabled, Preserve Shape produces a surface whose CVs coincide with the CVs of the cross-sections (after they have been converted to a common type and an identical number of CVs). The skinning algorithm is faster with shape preservation OFF, but it lacks precision.

Skinning with shape preservation ON may produce unintuitive shapes when the cross-sections have many coincident CVs or are very close to each other. In this case try to jitter the CVs, vary the V Order (see below), or simply disable shape preservation.

Preserve shape is deactivated when doing bi-linear skinning.

V WRAP

This menu (menu: *Off, On, If primitive does*) setting determines whether the surface should be wrapped in the V parametric direction. The options are to open (Off), close (On), or inherit the closure type from the cross-sections. V Wrap is ignored when doing bilinear skinning.

USE V ORDER

Enables or disables the use of the V Order parameter. If the flag is OFF, the skinned surface is built as a cubic (order 4) in V, unless fewer than four cross-sections for an open V or 3 cross-section for a closed V are given. For example, if the input consists of two faces and the V Wrap flag is OFF, the surface will be linear in V (order 2). The status of the V Order flag is irrelevant when the faces or surfaces are all polygons or meshes respectively, and when doing a bilinear skin.

Here, cross-section refers either to a face or a surface boundary, depending on the types being skinned.

V ORDER

/orderv

Specifies the order of the skinned surface when the V Order flag is enabled. A NURB surface of order “n” can be constructed with at least n or $n-1$ cross-sections, depending on whether the surface is open or closed in V respectively. A Bézier surface of the same order can be constructed with at least $M*(n-1) + 1$ cross-sections if open, or $M*(n-1)$ cross-sections if closed. M is a non-negative, integer multiplier. The V order is ignored when the faces or surfaces are all polygons or meshes respectively, and when building a bilinear skin.

SKIN / N

Can optionally skin subgroups of n primitives or every n th primitive in a cyclical manner. Example: Assume there are six primitives numbered for 0 - 5, and $N = 2$. Then:

- a) Groups will generate 0-1 2-3 4-5
- b) Skipping will generate 0-2-6 and 1-3-5.

N determines the number of primitives to be either grouped or skipped. $N \geq 2$.

KEEP PRIMITIVES

Determines whether the input primitives will be preserved (On) or deleted from the output (Off).

OUTPUT POLYGONS

If set, this flag instructs the program to convert the skinned surface(s) to polygons if the surface type is "mesh".

I 18.4 INPUTS / GEOMETRY TYPES

Any face type. Polygons, Béziers, NURBS curve.

The first input accepts either U curves (if a V curve is provided for the second input), or U and V curves supplied in Groups. Input V (cross-section) curves if you have provided U curves for the first input.

I 18.5 SEE ALSO

- *Bridge OP* p. 463
- *Cap OP* p. 468
- *Primitive OP* p. 697
- *Rails OP* p. 713
- *Sweep OP* p. 791

I 18.6 MOUSE AND KEYS -

-  Select items to be skinned in sequence. Type to be picked is dependent on the current *Select* Operation.
-  or  Used to select U/V cross sections for bilinear primitive skins.
-   Use to toggle the selection of an item.
-  Complete the skinning.
-   Pops-up the Operation menu.

118.7 DESCRIPTION -

Skinning can be used to skin points, edges, or primitives, based on the current selection type. If points, they are connected to form a polygon and all surface options specified in the Parameters Dialog are ignored. If edges, the effect is to create a “staircase” mesh. If primitives, you can only select faces (i.e. polygons or curves, but not parametric curves on surfaces). This is the case you are most likely to use on a regular basis.

You must **Shift** click on each item (cross-section) you want to skin or remove from the skin. If the faces are of different types (polygons, Bézier, NURBS) and/or have a different number of CVs, the operation brings them to a common denominator, always from a simpler type to a more complex type (e.g. polygon to NURBS) and from fewer to more CVs. It then skins the faces in the order in which they were selected to generate a surface.

To begin the skinning operation, select the curves by **Shift** **⌘** clicking them in the order they are to be skinned. To complete the skin, click anywhere in the viewport with the right mouse button (**⌘**), or type **Enter**.

BI-LINEAR SKINNING

If you are skinning primitives, you can build boundary surfaces by doing a bi-linear skin (i.e. skinning both in U and V). To do so, select the U cross-sections with the left mouse button (**⌘**) as described above, and the V cross-sections with the middle mouse button (**⌘**). You don't have to select all U faces first – the mouse button used will indicate which parametric direction you are adding the curve to. Don't forget to use **Shift** to extend the selection (or remove from it). You will notice that the first curve you click on turns into a surface. This is a simple way of turning a single boundary curve into a surface with interim points which can be further modeled.

CAPS ON A SKIN

To create end-caps on a skin, check the *Caps* button in the sub-icons. If enabled, end-caps are applied to both open and closed surfaces.

TIPS

- Select a curve, duplicate it with **Alt** **Shift** **D**, and **Alt** drag it several times to quickly create a framework for a skin operation.
- To unskin a portion of a surface currently in the process of being skinned, **Shift** click any of the curves to remove them from the skin.
- Before clicking the right mouse button (**⌘**) or **Enter** to complete the skin, you can change the skinning parameters in the Parameters dialog box and see the skin “recook”.
- If the cross-sections don't want to be selected, ensure you are currently selecting primitives (type **S** **3**) as a short-cut for entering the Selection Operation and choosing primitives).

118.8 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog for this Operation. See *Parameters – Skin Page* - p. 755.

KEEP PRIMITIVES

Determines whether or not the original primitives used to create the skin are kept after a skinning operation. If not, the original primitives are deleted.

118.9 SKIN OPERATION MENU -

Call up the menu for this Operation by using **Ctrl** .

FINISH SKINNING **Enter**

Completes the skinning operation.

118.10 PARAMETERS – SKIN PAGE -

CONNECTIVITY

- | | |
|--|--|
| <ul style="list-style-type: none"> • Rows • Columns • Rows & Cols
 • Triangles • Quadrilaterals • Alternating Triangles | <p>Creates horizontal lines.</p> <p>Creates vertical lines.</p> <p>Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon). Compare them in the Model Editor.</p> <p>Build the grid with Triangles.</p> <p>Generates sides composed of quadrilaterals (default).</p> <p>Generates triangles that are opposed; similar to the Triangles option.</p> |
|--|--|

PRESERVE SHAPE

Instructs the program to compute a skin that goes through all cross-sections (on) or just approximate them (off). A perfect shape will take longer to compute and might turn out less fair (more convoluted) than an approximated shape.

V WRAP

When enabled, creates a closed surface in the V direction.

USE V ORDER

Enables the ability to set the V order of the generated surface if the surface is NURBS or Bézier. If not checked, Houdini will attempt to build a cubic surface unless fewer than four cross-sections are given, in which case the V order will equal the number of cross-sections. If checked, skinning uses the V Order specified below.

V ORDER

Specifies the V spline order of a NURBS or Bézier skin. Enter the V order explicitly in this field. To build a valid spline surface of a given order, you need to have at least *order* cross-sections if the surface is open in V, or *order-1* cross-sections if the surface is wrapped in V.

118.11 PARAMETERS – CAPS PAGE -**FIRST U / V CAP**

Choose type of top cap to use: *No End Cap*, *End Cap Faceted* (sharp edges, vertices not shared at cap), *End Cap Shared* (smooth edges, vertices shared at caps), or *End Cap Rounded* (adds cross-sections to end).

LAST U / V CAP

Choose type of bottom cap to use: *No End Cap*, *End Cap Faceted* (sharp edges, vertices not shared at cap), *End Cap Shared* (smooth edges, vertices shared at caps), or *End Cap Rounded* (adds cross-sections to end).

DIVISIONS

Determines the number of divisions to add if the tube is created from polygons or a mesh. Use the slider to enter values quickly with the mouse.

I 19 SMOOTH OP

I 19.1 DESCRIPTION

The Smooth operation smoothes out (or "relaxes") polygons, meshes and curves, without increasing the number of points. It rearranges the existing points in the geometry to remove roughness.

Smooth can also smooth floating point attributes that are of size 1, 2 or 3, including colour and texture coordinate.

If there is an ambiguity between using a vertex attribute or a point attribute, the point attribute is used.

I 19.2 PARAMETERS

GROUP

Subset of primitives to smooth.

APPLY TO

Apply smoothing operation to point position, texture uv's, color or another attribute specified by name.

ATTRIBUTE NAME */attribname*

Point or vertex attribute to smooth.

CUTOFF FREQUENCY */frequency*

Noise frequency to remove.

The larger this value, the more it will keep the original shape of the geometry.

SMOOTHING ITERATIONS */iterations*

Number of smoothing steps.

I 19.3 SEE ALSO

- *Divide OP* p. 549
- *Fit OP* p. 577
- *SubDivide OP* - p. 777

I20 SOFT PEAK OP

I20.1 DESCRIPTION

Soft Peak translates the Source points along their normals. Unlike the Peak operation, Soft Peak takes into account a rolloff function and a radius of influence. The points within radius distance to the selected group will be displaced by an amount proportional to the distance.

I20.2 PARAMETERS

GROUP

Subset of points to translate.

DISTANCE */dist*

Distance along normal by which to translate

SOFT RADIUS */radius*

Area of influence

SOFT TYPE

Type of rolloff function

TANGENT ANGLES */tandeg*

Angles of the cubic rolloff function's tangents. The first value applies to the tangent farthest from the source point, the second applies to the tangent closest to the source point.

KERNEL FUNCTION

Metaball kernel to use when rolloff Type is "Meta-ball"

IGNORE POINT CONNECTIVITY

Soft-displace only points connected to the points in the group, or all points within the radius of influence

TRANSLATE ALONG LEAD NORMAL

Translate points within the radius along the normal of the point in Group closest to them. Otherwise translate along their respective normals, giving a mushrooming effect.

RECOMPUTE POINT NORMALS

Recomputes point normals if they exist.

I20.3 LOCAL VARIABLES

NPT	The number of points.
NPR	The number of primitives.

I20.4 SEE ALSO

- *Peak OP* p. 666
- *Soft Transform OP* p. 760
- *Transform OP* - p. 805
- *Clay OP* p. 498
- *Point OP* p. 667

I21 SOFT TRANSFORM OP

I21.1 DESCRIPTION

Soft Transform transforms the source points in "object space". Unlike the Transform operation, Soft Transform takes into account a rolloff function and a radius of influence. The points within radius distance to the selected group will be displaced by an amount proportional to the distance.

PARAMETERS

GROUP

Subset of points to transform.

TRANSFORM ORDER

Order in which transformations occur.

ROTATE ORDER

Order in which rotations occur.

TRANSLATE */tx /ty /tz*

Amount of translation along xyz axes.

ROTATE */rx /ry /rz*

Amount of rotation about xyz axes.

SCALE */sx /sy /sz*

Non-uniform scaling along XYZ axes.

PIVOT */px /py /pz*

Local pivot point for transformations.

SOFT RADIUS */radius*

Area of influence.

SOFT TYPE

Type of rolloff function.

TANGENT ANGLES */tandeg*

Angles of the cubic rolloff function's tangents. The first value applies to the tangent farthest from the source point, the second applies to the tangent closest to the source point.

KERNEL FUNCTION

Metaball kernel to use when rolloff Type is "Meta-ball"

IGNORE POINT CONNECTIVITY

Affect only points connected to the points in the group, or all points within the radius.

RECOMPUTE POINT NORMALS

Recomputes point normals if they exist.

PRESERVE NORMAL LENGTH

Normal lengths remain unaffected.

I21.2 LOCAL VARIABLES

CEX, CEY, CEZ	The centroid of the input.
GCX, GCY, GCZ	The centroid of the input group.
XMIN, XMAX	The X extents of the bounding box of the input.
YMIN, YMAX	The Y extents of the bounding box of the input.
ZMIN, ZMAX	The Z extents of the bounding box of the input.
SIZEX, SIZEY, SIZEZ	The size of the bounding box of the input.

I21.3 SEE ALSO

- *Transform OP* - p. 805
- *Primitive OP* p. 697
- *Point OP* p. 667
- *Soft Peak OP* p. 758
- *Edit OP* p. 553

I22 SORT OP

I22.1 DESCRIPTION

The Sort OP allows you to sort points and primitives in different ways. Sometimes the primitives are arranged in the desired order, but the point order is not. There are many possible combinations.

Tip: To sort vertices, use the *Primitive OP* p. 697.

I22.2 PARAMETERS – POINT PAGE

POINT SORT

Sort the points in the input geometry, according to the following criteria:

<i>No Change</i>	No sorting is applied.
<i>By Vertex Order</i>	Order points in same order as vertices.
<i>By X / Y / Z</i>	Sort according to X ,Y or Z position.
<i>Reverse</i>	Reverse the point order.
<i>Random</i>	Randomize point order using the specified seed without changing the point positions.
<i>Shift</i>	Shift points by the amount t specified on the offset line.
<i>Proximity to Point</i>	Sort points by their proximity to the specified point.
<i>Along Vector</i>	Sorts points along either a user or object-defined vector.

SEED */pointseed*

The random seed when Point Sort is set to *Random*.

OFFSET */pointoffset*

Shift point order by the amount specified on the offset line. A value of two makes the third point become the first point. The first and second points are placed at the end, etc.

POINT */pointprox /pointproxxy /pointproxz*

The X, Y and Z coordinates to reference when sorting by proximity to point.

VECTOR OBJECT

Sort points along a vector defined by the object's transformation values.

VECTOR */pointdirx /pointdiry /pointdirz*

Allows you to specify a unique vector along which points can be sorted.

EXPRESSION */pointexpr /primexpr*

Specifies an expression to sort by. Each object is reordered so the elements who evaluated to a higher expression value will be found after those with a lower value. Elements with equal expression value maintain their relative position.

I22.3 PRIMITIVE PAGE**PRIMITIVE SORT**

Sort the primitives according to the following criteria:

<i>No Change</i>	No sorting is applied.
<i>By X /Y / Z</i>	Sort according to X ,Y or Z position.
<i>Reverse</i>	Reverse primitive order.
<i>Random</i>	Randomize primitive order using the specified seed without changing the primitive positions.
<i>Shift</i>	Shift primitives by the amount the specified on the offset line.
<i>Proximity to Point</i>	Sort primitives by their proximity to the specified point.
<i>Along Vector</i>	Sorts primitives along either a user or object-defined vector.

SEED */primseed*

Random seed when sorting by *Random*.

OFFSET */primoffset*

Shift primitive order by the amount specified on the offset line. A value of two makes the third primitive become the first primitive. The first and second primitives are placed at the end, etc.

POINT */primproxx /primproxxy /primproxz*

The X, Y and Z coordinates to reference when sorting by *Proximity to Point*.

VECTOR OBJECT

Sort primitives along a vector defined by the object's translation.

VECTOR */primdirx /primdiry /primdirz*

Allows you to specify a unique vector along which primitives can be sorted.

I22.4 INPUTS / GEOMETRY TYPES

INPUT 1 – SOURCE DATA

Any geometry type.

I22.5 USES / WORKS IN RELATION WITH

- Allow points and primitives to be sorted in different ways.
- Sorting along a vector aids with transparency viewing in GL.

I23 SPHERE OP

I23.1 DESCRIPTION

This OP generates spherical objects of different geometry types. It is capable of creating non-uniform scalable spheres of all geometry types.

If an input is provided, the sphere's radius is automatically determined as a function of the input's bounding geometry.

I23.2 PARAMETERS

PRIMITIVE TYPE

Select from the following types. For information on the different types, see the *Geometry Types* section. Depending on the primitive type chosen, some OP options may not apply.

- Primitive
- Polygon
- Mesh
- NURBS
- Bézier

CONNECTIVITY

This option is used to select the type of surface, when using a *Mesh Primitive Type*.

- Rows Creates horizontal lines.
- Columns Creates vertical lines.
- Rows & Cols Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon).
- Triangles Build the grid with Triangles.
- Quadrilaterals Generates sides composed of quadrilaterals (default).
- Alternating Triangles Generates triangles that are opposed; similar to the Triangles option.

RADIUS */radx /rady /rad*

The radius of the sphere in X, Y and Z.

CENTER */tx /ty /tz*

Offset of sphere center from object center.

ORIENTATION

Determines axis for sphere. Poles of sphere align with orientation axis.

FREQUENCY */freq*

This controls the level of polygons used to create the sphere, when using the *Polygon Primitive Type*.

ROWS / COLUMNS */rows /cols*

Number of rows and columns in a sphere when using the mesh, imperfect NURBS and imperfect *Bézier*.

U / V ORDER

If a spline curve is selected, it is built at this order for U and V.

IMPERFECT

This option applies only to *Bézier* and NURBS spheres. If selected, the spheres are approximated nonrational curves, otherwise they are perfect rational curves.

UNIQUE POINTS PER POLE

Applies to *Mesh*, NURBS and *Bézier* surfaces only. This option determines whether points at the poles are shared or are individual to the columns.

I23.3 GEOMETRY TYPES

Primitive, Polygon, Mesh, NURBS, *Bézier*.

I23.4 MOUSE AND KEYS -

Ctrl 

Pops-up the Operation menu.

I23.5 DESCRIPTION -

This Operation is used to create spheres and ellipsoids. Clicking and dragging the mouse on the Construction Plane generates a sphere whose radii are specified by your drag.

Clicking the mouse button on the Construction Plane without dragging places a sphere with radii as specified in the Parameters dialog box (default of 1) at the location of the mouse click. The X and Y radii of the sphere are aligned with the X and Y axes of the Construction Plane.

Typing **Enter** places a sphere or ellipsoid whose size and position are specified in the Parameters dialog.

If an odd aspect ratio was previously entered in the Parameters dialog, clicking and dragging will produce ellipsoids which maintain that aspect ratio. This can be reset by clicking on the *Reset Radii* button.

123.6 SUB-ICONS -

PRIMITIVE / POLYGON / MESH / NURBS / BÉZIER

Click these buttons to select which type of sphere you want to draw. For differences between the five types, see *Geometry Types* section.

RESET RADII

Changes the radii of the sphere in the parameters dialog back to 1.

123.7 ELLIPSOID OPERATION MENU -

Call up the menu for this Operation by using **Ctrl** .

PRIM / POLY / MESH / NURBS / BÉZIER ELLIPSOID **E** / **P** / **M** / **N** / **B**

These five menu items allow you to specify whether you want to create an ellipsoid as a primitive, polygonal set, mesh, NURBS or Bézier surface.

BUILD DEFAULT ELLIPSOID **Enter**

Places an ellipsoid at the place currently specified by the center / radius settings in the Parameters dialog. The X and Y radii of the sphere are aligned with the X and Y axes of the Construction Plane.

123.8 OPERATION PARAMETERS -

CONNECTIVITY

- Rows Creates horizontal lines.
- Columns Creates vertical lines.
- Rows & Cols Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon).
- Triangles Build the grid with Triangles.
- Quadrilaterals Generates sides composed of quadrilaterals (default).
- Alternating Triangles Generates triangles that are opposed; similar to the Triangles option.

RADIUS

This is the X radius of an ellipsoid that is placed if you click on the Construction Plane without dragging. If you click and drag, the size of the sphere is over-ridden by the amount of drag. Entering non-equal values in the XYZ fields results in ellipsoidal shapes.

The X radius is defined by the distance dragged from the centre, while the Y and Z radii vary proportionally with the X / Y and X / Z ratios in the parameter dialog.

CENTER

Determines the location of the center of the ellipsoid. This value is updated whenever you click (and drag) to create an ellipsoid. A new ellipsoid centre will be positioned here if you hit **Enter**.

FREQUENCY

Increases or decreases the number of polygons which make up a polygonal sphere. The higher the frequency, the smoother the sphere. It is disabled if building a sphere of a type other than polygonal.

ROW / COLUMNS

The number of rows and columns of a mesh or imperfect NURBS / Bézier sphere. The more rows and columns, the rounder the sphere. A NURBS or Bézier sphere should have at least *order-1* rows and columns. Rows are associated with the V directions and columns with the U parametric direction.

U ORDER / V ORDER

Sets the U and V spline order of the NURBS or Bézier surface when building a sphere of one of these two types. The lowest order is 2 (linear); the highest is 11. Cubic spheres are built by default.

IMPERFECT

Specifies whether the NURBS / Bézier sphere should be built using rational or non-rational splines. A perfect sphere has a rational topology – one that associates non-unit weights with (some) of its vertices. Furthermore, a perfect sphere has a predefined number and positions of CVs for any given spline order. An imperfect sphere is non-rational and its number of CVs isn't that strictly determined by its order.

Rational spheres built this way yield a mathematically perfect shape; however, given their special definition, perfect spheres are not always the ideal choice for further modeling of their points. Besides, they represent heavier geometry and may put more pressure both on the CPU and RAM. In practice, you will find imperfect spheres to be a better modelling choice, so it is advisable to build perfect spheres only when perfect shapes are paramount.

UNIQUE POINTS PER POLE

In a mesh-type sphere, the meridians meet at the poles of the sphere. This creates a situation where each meridian line contributes its own point to a pole. When this box is *not* checked, the points are consolidated into a single point shared by all the meridians. Otherwise if checked, the points are all left to be unique.

I24 SPRING OP

I24.1 DESCRIPTION

Deforming and moving the input geometry using “forces” instead of channels.

Geometry is deformed using forces that simulate the laws of physics on the points. A simulated “mass” is assigned to each point. Its primitive edges act as “springs” which oppose the forces, and pull the points back toward their original positions. When the springs are stretched by the forces, they try to pull the points back. The points do not stop moving when they return to their original positions, however, but continue to oscillate because of their mass, until the oscillation dies out.

Forces which act upon the points are as follows:

- external force (gravity)
- wind (similar to external force)
- turbulence (chaotic forces)

The greater the drag value, or smaller the mass, the faster the oscillation dies out.

Note: The Spring OP is fine for many situations, but for any more sophisticated simulations, you should setup the simulation using the RBD Operation – this allows much better control using POPs (for simulating the dynamics), and CHOPs (to record the simulation data and play it back more quickly). The *Physical* page of geometry Objects allows you to control the properties of an object in such a simulation.

I24.2 PARAMETERS – STATE PAGE

START TIME */timestart*

The start time is the time at which the simulation starts cooking. Before this, the simulation will be static. If set to some later time, the points will not move until that time is reached. The default is *Tstart* (frame one).

PREROLL TIME */timepreroll*

How many seconds of the simulation to bypass, after the reset time is reached. For example, if you put the number 33 into this field (and reset is at *Tstart*), frame one will show the simulation that was at a time of 33 seconds. In other words, the first thirty-two seconds have been bypassed, and the time at thirty-three seconds is shifted to frame one. The first thirty-two seconds must still be calculated in order to compute the status of the points, so you will notice some delay upon reset.

TIME INC */timeinc*

The *Time Inc* parameter determines how often to cook the OP. By default, this parameter is set to $1/\$FPS$. This means that the OP will cook once for every frame. When complex dynamics are involved, the OP may require more frequent cooking for increased mathematical accuracy. To get sub-frame accuracy in the cooking, set the *Time Inc* to something smaller than $1/\$FPS$. For example, setting the *Time Inc* to $0.5/\$FPS$ will mean that the OP gets cooked twice for every frame.

■ **Note:** Never set this parameter greater than $1/\$FPS$.

ACCURATE MOVES

This option makes the nodes move more accurately between frames by calculating their trajectories for fractional frame values.

ATTRACTOR USE

All Points

All point attractors affect all particles (or points).

Single point per particle

When enabled, each particle is assigned a single attractor point, and is affected by only that point. Assignment is done by point number modulo the total number of attractor points.

I24.3 PARAMETERS – FORCES PAGE**EXTERNAL FORCE** */externalx /externaly /externalz*

Forces of gravity acting on the points. When drag is zero, the points can accelerate with no limit on their speed.

WIND */windx /windy /windz*

Wind forces acting on the points. Similar to external force. Using wind (and no other forces, such as turbulence), the points will not exceed the wind velocity.

TURBULENCE */turbx /turby /turbz*

The amplitude of turbulent (chaotic) forces along each axis. Use positive values, if any.

TURB PERIOD */period*

A small period means that the turbulence varies quickly over a small area, while a larger value will cause points close to each other to be affected similarly.

SEED */seed*

Random number seed for the simulation.

124.4 PARAMETERS – NODES PAGE

FIXED POINTS

This is a point group. All points in the point group will remain unaffected by the forces. Also see the *Group OP* p. 592 for notes on how to specify point ranges.

FIXED POINTS GO TO SOURCE POSITIONS

Determines whether or not points in the *Fixed Points* group should be moved to the positions of the corresponding points in the source geometry.

COPY GROUPS FROM SOURCE

Determines if the Spring OP should copy groups from the Source geometry at each frame. This lets you specify the name of an animating group in the *Fixed Points* field, and the contents of this group will be kept up to date.

ADD MASS ATTRIBUTE

When selected, the mass is computed for the deforming geometry.

MASS */mass*

Mass of each point. Heavier points take longer to get into motion, and longer to stop.

ADD DRAG ATTRIBUTE

When selected, the geometry is deformed by the drag attribute.

DRAG */drag*

Drag of each point.

SPRING BEHAVIOUR

How the springs will behave:

Hooke's Law Springs will work according to Hooke's Law. This is generally more stable than *Normalized Displacement*.
Hooke's Law: $Force = Displacement \times Spring\ constant$

Normalize Displacement Similar to Hooke's law except that the displacement is normalized to the original length of the Spring (this was the behaviour in prior releases – 2.5 and earlier).

SPRING CONSTANT */springk*

The spring constant. How tight the springs are. Increase this value to make the springs tighter and thus make the object more rigid. As this number becomes higher, the springs can oscillate out of control. Decrease the *Time Inc* if this happens.

INITIAL TENSION */tension*

The Initial k constant of the geometry before being deformed by the spring operation.

I24.5 PARAMETERS – LIMITS PAGE**+ / - LIMIT PLANE** */limitposx ... /limitposz /limitnegx ... /limitnegz*

The points will bounce off the limit planes when it reaches them. The six limit plane fields define a bounding cube. The default settings are one thousand units away, which is very large. Reduce the values to about one to see the effect.

HIT BEHAVIOR

Control over what happens when the geometry hits either the six collision planes or the collision object. The options are:

Bounce on Contact Geometry will bounce upon contact with the Collision input.

Stick on Contact Geometry will stick to the Collision input upon contact.

GAIN TANGENT / NORMAL */gaintan /gainnorm*

Friction parameters which can be regarded as energy-loss upon collision. The first parameter affects the energy loss (gain) perpendicular to the surface. 0 means all energy (velocity) is lost, 1 means no energy is lost perpendicular to surface. The second parameter is the energy gain tangent to the surface.

I24.6 INPUTS / GEOMETRY TYPES**SPRING SOURCE**

Any geometry that provides points, e.g. polygon sphere, mesh.

COLLISION OBJECT

This input defines an object for the nodes to collide with. When this happens, the geometry can either stick or bounce. It is important to note that when the collision object is deforming, collision detection may fail, causing some portions of the geometry to “leak through” the collision object.

FORCE

The Force input acts accepts input from a Force OP which uses a metaball shape as a force field allowing forces to be defined in vortices or accelerated along an axis. Refer to the *Force OP* p. 584 for further explanation.

Note: The Spring OP will use point normals as initial node velocity if point normal attributes exist *and* there are no point velocity attributes in the incoming data. If you add velocity attributes to the points, the point normals are ignored.

124.7 LOCAL VARIABLES

none.

124.8 USES / WORKS IN RELATION WITH

Deforming and moving the Source geometry using “forces” instead of channels.

124.9 SEE ALSO

- RBD Operation.
- Objects > *Physical* page.
- POPs section of Reference manual.
- *Force OP* p. 584
- *Metaball OP* p. 638
- *Particle OP* p. 653

Starburst insets points on polyonal faces. It can be used for controlled subdivision along the edges of connected polygons.

I25 STARBURST OP

I25.1 DESCRIPTION

Starburst insets points on polyonal faces. It can be used for controlled subdivision along the edges of connected polygons.

I25.2 PARAMETERS

GROUP

Points to inset.

INSET */inset*

The amount by which to inset the point.

DISTANCE */dist*

The amount by which to translate the points in Group along their normals.

I25.3 LOCAL VARIABLES

NPT The number of points.

NPR The number of primitives.

I25.4 SEE ALSO

- *PolyExtrude OP* p. 674
- *SubDivide OP* - p. 777

I26 STITCH OP

I26.1 DESCRIPTION

The Stitch OP is used to stretch two curves or surfaces to cover a smooth area. It can also be used to create certain types of upholstered fabrics such as cushions and parachutes.

If a second input is given, it must contain one surface that the primitives in the first input can stitch to. The left input can contain either faces or surfaces; in either case, each primitive in the first input is stitched to a parametric area of the surface in the second input in such a way that the parametric area allocated to each primitive is the same and the size of all areas added together equals the parametric range specified in the *R Width* (see below).

Please refer to the *Align OP* p. 430 for a discussion of “left” and “right” primitives as well as the option of an auxiliary input.

I26.2 PARAMETERS

Please refer to the *Align OP* p. 430 for a discussion of “left” and “right” primitives, which are used in the descriptions below.

GROUP

Which primitives to stitch. If blank, it stitches the entire input. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

STITCH */ n*

Optionally stitches sub-groups of *n* primitives or patterns of primitives. The value entered for *N* determines the pattern of primitives stitched.

WRAP LAST TO FIRST

If enabled, it connects the beginning of the first primitive in the left input to the end of the last primitive in the same input. If only one primitive exists, its ends will be stitched together.

DIRECTION

Allows stitching along either the U or V parametric direction.

TOLERANCE */tolerance*

This parameter minimizes modification to the input sources. A smaller value creates less modification.

BIAS */bias*

Determines which primitive remains unaffected. The values go from 0 - 1, where 0 – first, and 1 – last.

LEFT UV / RIGHT UV *//leftuv1 //leftuv2 /rightuv1 /rightuv2*

Point on each left / right primitive at which to begin / end the stitch.

LR WIDTH *//rwidth1 //rwidth2*

The first value represents the width of the left stitch. The second value represents the width of the right stitch.

STITCH

If selected, move a single row from each primitive to coincide.

TANGENT

If selected, modifies neighbouring rows on each primitive to create identical slopes at the given rows.

SHARP PARTIALS

If selected, creates sharp corners at the ends of the stitch when the stitch partially spans a primitive.

FIXED INTERSECTION

When the tangent option is on, this option allows some flexibility as to which side of each slope is modified.

LR SCALE *//rscale1 //rscale2*

Use this parameter to control the direction and position of the tangential slopes.

126.3 SEE ALSO

- *Fillet OP* p. 574
- *Join OP* - p. 602

I27 SUBDIVIDE OP -

I27.1 DESCRIPTION

This OP takes an input polygon surface (which can be piped into one or both inputs), and divides each face to create a smoothed polygon surface using a Catmull-Clark subdivision algorithm. It is similar to the Paste OP in that it divides up all or part of a surface allowing you to increase areas of local detail – especially useful for avoiding the angular appearance often associated with polygonal models – without adding lots of extra geometry to the entire object. While the topology of the input mesh can be arbitrary, for best results all polygons should be convex and relatively uniform in distribution.

All polygons in the left input which are specified by the "Group" field are used to determine the polygon mesh to subdivide. For polygon edges to be classified as the same edge, they must share the same points. Merely being physically close is not sufficient.

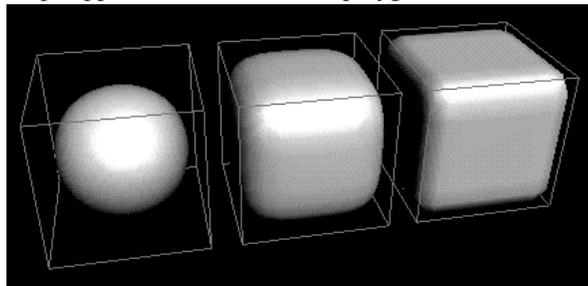
The elements of the right input specified by the "Creases" field are used as creases. Each edge in a crease polygon corresponds to the edge in the polygon mesh which has the same point numbers. Again, point position is irrelevant.

Vertex attributes are supported and preserved, however, instead of being properly subdivided they are merely linearly interpolated.

CREASES

Creases control the strength of pull of the polygon faces on the subdivision surfaces, much like a magnet drawing the surface towards the reference polygon. They can be applied selectively using the Creases field to specify an input group to use.

Creases work by controlling the strength of the pull of the polygon faces on the subdivision surfaces, like a magnet drawing the surface towards the reference polygon. The figure below shows the result of setting the *Crease Weight* to 0, 1, 2 reading from left to right. As the weight increases so the pull effect strengthens and the shape approaches the reference polygon.



CREASE WEIGHTS AND RENDERMAN

Crease weight attributes which are added, but not yet Subdivided are preserved when a RIB file is generated. They are then used internally by RenderMan to implement subdivision surfaces during the rendering process.

Note: In order to output the crease weights in the RIB file, your geo object must have the *Render* page > *Geometry* > *Polygons as Subdivision Surfaces* parameter set.

THE CREASE ALGORITHM

If there is a second input:

- If the *Override* button is enabled, each edge defined in the second input will have its edge crease weight set to the value of the override.
- If the vertex attribute "creaseweight" exists on the second input, each matching edge in the input will have its crease weight set to the maximum of the vertex attributes for any shared edges.
- If the primitive attribute "creaseweight" exists on the second input, each polygon in the second input will set matching edges to the appropriate weights.

If there is no second input:

- If an override is specified, the value of the override is used for all edges in the sub-divided surface.
- If the vertex attribute "creaseweight" exists, this attribute will be used to define the crease weights on the edges of the surface.
- If the primitive attribute "creaseweight" exists, this attribute will be used to define the crease weights for the subdivision surface.

When defining crease weights on shared edges, the maximum of the weights of the shared edges is used. For example, if two polygons share an edge and the primitive attribute is used, the maximum of the crease weight will be used for the shared edge.

CLOSING CRACKS

Cracks can be closed by either Pulling or Stitching, so only one of these two options can be chosen at a time from the *Surrounding Faces* parameter. *Bias* applies only to Pulling, and is disabled when Stitching is chosen.

A crack is formed by a single edge on the non-subdivided area and multiple edges on the subdivided area. The 'Surrounding Faces' menu determines what will happen to the single edge on the non-subdivided area, and in more generally, to the polygon containing this edge.

If *No Edge Division* is chosen and cracks are pulled closed, all the points on the subdivided edges are pulled (i.e. moved) to the closest points on the non-subdivided edges. *Bias* is disabled, when *No Edge Division* is specified.

If cracks are pulled with the *Divide Edges* option, the non-subdivided edge is split into many sections, so that each point on the non-subdivided edge now corresponds to a new point on the subdivided edge. Then, points on the newly divided edge are joined with the points on the subdivided boundary. A *Bias* of 1 will place these joined points along the subdivided boundary. A *bias* of 0 will place them along

the non- subdivided boundary (and values between 0 and 1 will place them somewhere inbetween).

Pulling cracks with the *Triangulate* option will do exactly the same thing as *Divide Edges*, except it will also triangulate the non-subdivided polygon. This is desirable because pulling the non-subdivided boundary towards the curved subdivided boundary will likely generate a non-planar polygon, so *Triangulate* will divide this polygon into smaller (planar) ones. Pulling cracks with a Bias of 1 and triangulating usually produces the nicest results.

The *Triangulate* option is necessary because the Divide OP is not designed to handle (very) non-planar polygons.

The *Stitch Cracks Together* option, on the other hand, inserts new polygons (triangles) to close up the cracks. When *No Edge Division* is chosen, Many triangles are created, each having one vertex on one point of the non-subdivided edge.

When *Divide Boundary Edges* is chosen, the non-subdivided edge is divided, so there are more points available to be used for triangles. The resulting triangles are more regularly shaped (not as long and skinny). The triangulate option will again triangulate the non-subdivided polygon, although this option is less likely to be used because this polygon should remain planar during stitching.

127.2 PARAMETERS

GROUP

Subset of input to use as a polygonal mesh to subdivide. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

CREASES

This field allows you to specify a subset of the second input to use as creases. The elements of the second input specified by the *Creases* field are used as creases. Each edge in a crease polygon corresponds to the edge in the polygon mesh which has the same point numbers. Point position is irrelevant. For polygon edges to be classified as the same edge, they must share the same points. Merely being physically close is not sufficient.

The primitive attribute *creasesharpness* is used to determine the sharpness of the specified creases unless overridden.

edge groups

You can also specify edge groups for creases. The edges can also be brought in from a parent's edge selection.

The syntax for specifying edges is as follows: p1-2 indicates an edge between points 1 and 2 p1-2-3 indicates two edges, one from point 1 to 2 and one from 2 to 3... and so on. 0e1 indicates edge 1 on primitive 0. 3 indicates all the edges of primitive 3 group1 indicates all the edges of all the primitives in the group group1.

DEPTH */iterations*

How many iterations to subdivide. Higher numbers give a smoother surface.

OVERRIDE CREASE WEIGHT ATTRIBUTE */overridecrease*

Determine if the crease sharpness should be determined by the primitive or vertex “creaseweight” attribute or by overridden by this OP.

CREASE WEIGHT */creaseweight*

If the crease weight is overridden, this is the weight used.

Tip: The default is to have the *Override Crease Weight Attribute* enabled. So you can simply set a value which applies to all the creases. You can, however, set a crease attribute using the Vertex or Primitive OPs which allows for more control.

GENERATE RESULTING CREASES */outputcrease*

If any creases are sharper than the *Depth*, they will be left over in the resulting geometry. With this parameter enabled, these left over creases are created with the appropriate primitive attribute values, and placed in the specified group.

NEW GROUP */outcreasegroup*

Name of the group to place the generated creases into.

CLOSE CRACKS

<i>Do Not Close</i>	Don't close cracks.
<i>Pull Cracks Closed</i>	Move points on boundary of subdivided area in order to close cracks formed during the subdivision.
<i>Stitch Cracks Together</i>	Add polygons (triangles) to close the cracks caused by subdividing.

BOUNDARY

<i>No Edge Division</i>	When <i>No Edge Division</i> is chosen, many triangles are created, each having one vertex on one point of the non-subdivided edge.
<i>Divide Edges</i>	Divides edges surrounding the subdivided area when pulling or stitching cracks by inserting new polygons (triangles) to close up the cracks.

Triangulate

Does exactly the same thing as *Divide Edges*, except it also triangulates the non-subdivided polygon. This is desirable because pulling the non-subdivided boundary towards the curved subdivided boundary will likely generate a non-planar polygon, so *Triangulate* will divide this polygon into smaller (planar) ones. Pulling cracks with a *Bias* of 1 and triangulating usually produces the nicest results.

BIAS

/bias

Determines which points are moved when pulling crack closed.
 0 means move points on subdivided area to meet boundary.
 1 means move points on boundary to meet subdivided area.

I27.3 SEE ALSO

- *Divide OP* p. 549

I27.4 MOUSE AND KEYS -

-  Select faces.
-  Select edges.
-  or **Enter** Perform subdivision on selected faces.
- Ctrl**  Pops-up the Operation menu.

I27.5 DESCRIPTION -

This Operation takes a polygonal surface and divides each curve to create a smoothed polygon surface using a Catmull-Clark subdivision algorithm. It is similar to the Paste OP in that it divides up all or part of a surface allowing you to increase areas of local detail – especially useful for avoiding the angular appearance often associated with polygonal models – without adding lots of extra geometry to the entire object. You could also compare it to the Divide OP’s *Smooth* function, which is a quadratic method similar to the cubic method employed here. For best results, polygons should be convex and relatively uniform in distribution.

-  Selects the polygons to be subdivided. Box selections are allowed. These polygons are shown in yellow.
-  Selects edges. Box selections of edges are also possible. Selected edges are displayed with dashed red lines.
-  or **Enter** Performs the subdivision.

SETTING CREASE WEIGHTS

Creases control the strength of pull of the polygon faces on the subdivision surfaces, much like a magnet drawing the surface towards the reference polygon. You should generally use the *Crease OP* p. 524 to set your creases.

Alternately, within the SubDivide OP, no set crease weights for edges, first select the edges. Then, modify the value of the *Crease Weight* parameter. The weight is changed for all selected edges. A weight of zero specifies no creases. Edges with non-zero weights are displayed in dotted cyan lines.

To display the crease weight for an edge, simply select it and the Crease Weight parameter displays the current value. If multiple edges are selected and they all have the same weight, this weight will be displayed. If multiple edges are selected and their weights are not all the same, a "--" is displayed. Entering a value at this point will set the weights of all selected edges to the new value.

Crease weights are cleared after a subdivision. They are stored as a vertex attribute, so crease weights set in a Vertex OP can be used as an input to the Subdivide Operation. Crease weights set in the modeller can later be used in the Subdivide OP.

interactively setting crease weights

It is possible to interactively set crease weights and look at how the subdivided output changes. To do this, append a Subdivide OP after the OP of the surface being subdivided. Select the desired *Depth* and be sure to turn off *Override Crease Weight Attribute*. Enable the template flag for this OP.

Then, go into the Model Editor on the OP (by selecting from the OP tile's pop-up menu) immediately above the Subdivide OP. You can then modify the crease weight values of specific edges and interactively view the changes in the output of the Subdivide OP.

crease weights and renderman

Crease weight attributes which are added, but not yet Subdivided are preserved when a RIB file is generated. They are then used internally by RenderMan to implement subdivision surfaces during the rendering process.

Note: In order to output the crease weights in the RIB file, your geo object must have the *Render* page > *Geometry* > *Polygons as Subdivision Surfaces* parameter set.

127.6 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog, see *Operation Parameters* - p. 568.

CREASE WEIGHT

Creases control the strength of pull of the polygon faces on the subdivision surfaces, much like a magnet drawing the surface towards the reference polygon.

DEPTH

How many iterations to subdivide. Higher numbers yield a smoother surface.

I27.7 SUBDIVISION OPERATION MENU -

SELECT ALL POLYGONS / EDGES **A** / **Shift****A**

Selects all polygons / edges.

SELECT ALL POLYGONS/EDGES IN EDGE/POLYGON SELECTION **L** / **Shift****L**

Selects all polygons/edges within a selection.

TOGGLE POLYGON / EDGE SELECTION **T** / **Shift****T**

Toggles the selection of polygons/edges on/off.

CLEAR CREASE WEIGHT FOR SELECTED / ALL EDGES **R** / **Shift****R**

Clears crease weights on selected/all edges so they return to the default crease weight (specified in $\oplus > Crease Weight$).

I27.8 OPERATION PARAMETERS -

CLOSE HOLES

- | | |
|------------------------------|---|
| <i>Do Not Close</i> | Don't close holes. |
| <i>Pull Holes Closed</i> | Move points on boundary of subdivided area in order to close holes formed during the subdivision. |
| <i>Stitch Holes Together</i> | Add polygons (triangles) to close the holes caused by subdividing. |

SURROUNDING FACES

- | | |
|-------------------------|--|
| <i>No Edge Division</i> | When <i>No Edge Division</i> is chosen, many triangles are created, each having one vertex on one point of the non-subdivided edge. |
| <i>Divide Edges</i> | Divides edges surrounding the subdivided area when pulling or stitching holes by inserting new polygons (triangles) to close up the holes. |

Triangulate

Does exactly the same thing as *Divide Edges*, except it also triangulates the non-subdivided polygon. This is desirable because pulling the non-subdivided boundary towards the curved subdivided boundary will likely generate a non-planar polygon, so *Triangulate* will divide this polygon into smaller (planar) ones. Pulling holes with a *Bias* of 1 and triangulating usually produces the nicest results.

BIAS

Determines which points are moved when pulling hole closed.
0 means move points on subdivided area to meet boundary.
1 means move points on boundary to meet subdivided area.

127.9 PARAMETERS – DISPLAY PAGE -

DISPLAY COLOR

Use the color sliders to choose a color for the created bone objects.

Capturing and animation is much easier if the object colors are used to their full advantage. For example, in the Capture OP, the Capture Region colors are used to color the captured surface. When creating bones in the Bone Creation Operation, you can set the color of each bone that you create by using the color sliders as you're creating the bones.

127.10 PARAMETERS – EDIT BONES PAGE -

EDIT BONE

Specify a bone to edit by selecting from the pop-up menu.

REFERENCE FRAME

Specify world or object space. When set to world space, transformations will take place relative to objects; when set to object space, transformations will take place in a localised coordinate system within the object.

POSITION

Enter X, Y, and Z values here for the position of the bone. The coordinates are made relative to the Reference Frame specified (above).

I28 SUBNET OP

I28.1 DESCRIPTION

The Sub-net OP is essentially a way of creating a macro to represent a collection of OPs as a single OP in the Network Editor. The Sub-net OP can contain an entire OP Network within it, stream-lining and simplifying your OP network both visually and conceptually.

Selecting *Edit Sub-Network...* from the OP's pop-up menu presents you with a new Network Editor with four sub-network inputs. These four inputs are connected directly to the four inputs on the Sub-net OP in your original network. Proceed by attaching OPs as required to these four sub-network inputs. The display OP will be wired back to the output connector of the Sub-net OP in your original OP network. To get back to the original OP network, go up a level (type U).

Please refer to the *Sub-Networks* p. 190 in the *Interface* section for a complete discussion and an example of how to use sub-networks.

Tip: Select several Operators that you want to make into a sub-network, and select *Collapse Selected* from the OP's pop-up menu to automatically create a sub-network out of them. You will see the selected Operators replaced by a single Sub-network OP, and it will be properly rewired to contain the previously selected OPs.

I28.2 PARAMETERS

INPUT #1 - #4 LABEL

These labels are displayed when you click with F^{A} on one of the Sub-net OP's inputs. This is useful for remembering what the inputs are used for in your Sub-network.

I29 SUPERQUAD OP

I29.1 DESCRIPTION

Generates an isoquadric surface. This produces a spherical shape that is similar to a metaball, with the difference that it doesn't change its shape in response to what surrounds it. You can change the *XY Exponent* of an isoquadric surface to define it to be more "suarish" or "starish" in shape. Also, an isoquadric surface is always defined as a polygonal or mesh type geometry.

I29.2 PARAMETERS

PRIMITIVE TYPE

Select from the following types. For information on the different types, see the *Geometry Types* section. Depending on the primitive type chosen, some OP options may not apply.

- Polygon
- Mesh

CONNECTIVITY

This option is used to select the type of surface, when using a *Mesh* primitive type.

- Rows Creates horizontal lines.
- Columns Creates vertical lines.

- Rows & Cols Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon).
- Triangles Build the grid with Triangles.
- Quadrilaterals Generates sides composed of quadrilaterals (default).
- Alternating Triangles Generates triangles that are opposed; similar to the Triangles option.

RADIUS */radx /rady /radz*

Determines overall radius.

CENTER */tx /ty /tz*

Offset of superquad center from object center.

ORIENTATION

Determines pole axis for the iso surface.

ROWS / COLUMNS */rows /cols*

Number of rows and columns used in the superquad.

XY / Z EXPONENT */exp_{xy} /exp_z*

The XY exponent determines inflation / contraction in the X and Y axes. The Z exponent determines inflation / contraction in the Z axis. See the *Metaball OP* p. 638 for a description of exponents.

MULTIPLE POINTS PER POLE

Determines whether points at the poles are shared or are unique to the columns.

CUSP POLYGONS

Makes points unique, causing the superquad to be faceted.

CUSP ANGLE

Input angle in degrees to determine when vertices are shared or not, creating cusp-ing.

129.3 SEE ALSO

- *Metaball OP* p. 638

I30 SURFSECT OP

I30.1 DESCRIPTION

This OP performs boolean operations with NURBS and Bézier surfaces, or only generates profiles where the surfaces intersect. The individual surfaces do not need to be solids (i.e. wrap in U and V), nor do they need to form a solid as a group (for example, you can cut a grid with a sphere).

The surfaces in the first input are denoted by “A” in the parameter list. The surfaces in the second input are denoted by “B”. The entire A set is intersected with the B set, which allows for true CSG operations with spline surfaces. Thus, if A forms a solid and B forms a solid, any boolean operation between A and B will produce a solid. If either set has an open topology, the result will also be open.

What is deemed to be the inside and outside of a trim region depends on the direction of the surface normals. If necessary, use the Primitive OP to reverse the surface normals by reversing the surface’s U or V direction.

I30.2 PARAMETERS

GROUP A

Subset of NURBS and Bezier surfaces. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

GROUP B

Subset of NURBS and Bezier surfaces to intersect with A. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

3D TOLERANCE */tol3d*

World space precision of the intersection.

2D TOLERANCE */tol2d*

Domain precision of the intersection.

MARCHING STEPS */step*

Number of steps for tracing each profile span.

BOOLEAN OPTIONS

Enables boolean operations between the surfaces in A and B:

operation

Select from the following operations: *Union*, *Intersect*, *A-B*, *B-A*, or *User-defined*. If the Operation is set to *User-defined*, the following options become available:

<i>Keep Inside A</i>	Preserve the inside sections of the A surfaces.
<i>Keep Inside B</i>	Preserve the inside sections of the B surfaces.
<i>Keep Outside A</i>	Preserve the outside sections of the A surfaces.
<i>Keep Outside B</i>	Preserve the outside sections of the B surfaces.

RADIUS

Instead of always having a sharp edge, round the intersection edges by the specified radius. 0.0 gives previous behaviour.

SPAN DENSITY

How densely should the isoparms be made.
isoparms = length of fillet * density / radius.

CUSP ANGLE

At what angle the fillet is considered sharp and a sharp edge placed.

GENERATE PROFILES OPTIONS

Create profiles only where the surfaces intersect.

target

Which surface set to output profiles for: A, B, or both.

a profiles group

Place the A profiles in a user-defined group.

b profiles group

Place the B profiles in a user-defined group.

avoid already trimmed-out parts

Intersect only the visible surface parts and truncate the intersection profile at the trimmed-in surface boundaries.

join profiles created by multiple surfaces

If a surface has several adjacent profiles caused by its intersection with two or more surfaces, the profiles will be joined into a single curve-on-surface.

I30.3 SEE ALSO

- *Boolean OP* p. 456
- *Curvesect OP* p. 537

131 SWEEP OP

131.1 DESCRIPTION

The Sweep OP sweeps primitives in the Cross-section input along Backbone Source primitive(s), creating ribbon and tube-like shapes. The cross-section primitives are placed at each point of the backbone perpendicular to it. If a Reference Point input is given, each primitive will be oriented to aim at its corresponding reference point.

The Backbone Source can have one or several primitives. If there is more than one, Sweep will sweep the cross section along each one.

A backbone is a primitive curve that can be open or closed, but must have at least two points. The cross section input can also have multiple primitives, and can be assigned to the backbone in various ways. The origin of the cross section primitive is placed at a point on the backbone by default, but you can also choose a point number of the cross section to place. In most cases, it is best to build the cross section primitives in the XY plane; Sweep will automatically orient them properly along the backbone. If the backbone primitive(s) have point colors or texture coordinates, they will be maintained and applied to the cross section primitives.

131.2 PARAMETERS – SWEEP PAGE

X-SECTION */xgrp*

Primitives to use as cross section.

PATH GROUP */pathgrp*

Primitives to use as backbone.

REFERENCE GROUP */refgrp*

Points to used to the primitives.

CYCLE TYPE */cycle*

All Primitives at each point Places all the primitives from the Cross-section input at each point on the backbone.

One Primitive at a time Similar to the above, except the transformation is applied to individual primitives rather than to the whole.

Cycle Primitives This cycles through incoming primitives when placing them on a backbone. i.e. Start with 0 at vertex 0, primitive 1 > vertex 1, etc.

ANGLE FIX */angle*

Attempts to fix buckling twists that may occur when sweeping.

FIX FLIPPING */noflip*

Attempts to fix buckling twists that may occur when sweeping by fixing flipped normals.

REMOVE COINCIDENT POINTS ON PATH */skipcoin*

When selected, any points right on top of one another will be ignored.

AIM AT REFERENCE POINTS

Reference Points are used in conjunction with the backbone to control the orientation of the elements along the sweep. This is done by drawing a line between the reference point and corresponding backbone point in order to determine an angle vector which determines the orientation of the cross-section profiles. Enable this parameter to allow this behaviour.

Note! In order for this to work, you must supply Reference Points via the third input, and there must be exactly the same number of Reference Points as there are points in the Backbone.

USE VERTEX */usevtx*

Use vertex number of the incoming cross-section to place the cross-section on the backbone.

CONNECTION VERTEX */vertex*

Specify a specific vertex to connect to the backbone.

SCALE */scale*

Scales the cross sections globally.

TWIST */twist*

Cumulative rotation of the cross sections around the backbone. If a value of five is specified, the cross section at the first point of the backbone is rotated five degrees, the next ten degrees, the next fifteen degrees and so on.

ROLL */roll*

Non-cumulative rotation of the cross sections around the backbone. All cross sections get the same rotation.

131.3 PARAMETERS – OUTPUT PAGE

CREATE GROUPS */newg*

Selecting this option enables the creation of groups. A group is created for each backbone that is incoming. This allows for easy skinning in the Skin OP.

SWEEP GROUPS */sweepgrp*

Specify the name of your output groups in this field.

SKIN OUTPUT */skin*

On / Off Skins / doesn't skin the output.

On with Auto Close Closes the skinned mesh if the path curve which it follows is also closed. This allows you to close primitives properly.

On with Preserve Shape Like "On", but ensures the swept curves are fully encased in the resulting surface.

On with Preserve Shape and Auto Close Skin with "Preserve Shape" and "Auto Close" put together.

production tip

There is a way of speeding the skinning of many points using the second input of the Point OP. Suppose you have Thousands of procedurally animated curves you wish to skin with the Sweep OP – rather than performing a skinning operation after the animation, make a second set of unanimated geometry that is preskinned. Then assuming you have a matching number of points you can just swap in the animated points into the skinned geometry.

This technique is significantly faster than using the *Skin Output* option of the Sweep OP, or a Skin OP following the Sweep OP.

FAST SWEEP */fast*

Enables an optimized skinning technique which speeds up output from 2 - 4 times in many cases at the expense of accuracy. In order for it to work correctly, the input topologies must remain consistent between cooks and each cross-section must have the same number of vertices.

OUTPUT POLYGONS */polyout*

Outputs meshes as polygons rather than meshes.

131.4 LOCAL VARIABLES

<i>PT</i>	The current vertex number.
<i>NPT</i>	The number of points per detail.
<i>PATH</i>	The path primitive number.
<i>PCT</i>	Percentage along the backbone (path).

131.5 INPUTS

- Input 1 – The Cross-section
- Input 2 – The Backbone path
- Input 3 – Reference Points

131.6 SEE ALSO

- *Rails OP* p. 713
- *Skin OP* p. 749

I32 SWITCH OP

I32.1 DESCRIPTION

Allows switching between up to 9999 possible inputs. The output of this OP is specified by the *Select Input* field. This is useful for switching the output between several inputs depending on the evaluation of an expression placed in the *Select Input* field.

I32.2 PARAMETERS

SELECT INPUT */input*

The index of the source to use, where the first source is 0, the second 1, and so on.

I32.3 INPUTS / GEOMETRY TYPES

Multiple OP input – several OPs are connected to the input.

I32.4 LOCAL VARIABLES

none

I32.5 EXAMPLE

Below are two expressions that could be entered in the *Select Input* field:

$\$F-1$ The first input is selected at frame 1, the second at frame 2, etc.

$(\$F > 5)$ The first input is selected for the first five frames, then the second for the remaining frames.

I33 TORUS OP

I33.1 DESCRIPTION

Generates complete or specific sections of torus shapes (like a doughnut).

I33.2 PARAMETERS – TORUS FOLDER

PRIMITIVE TYPE

Select from the following types. For information on the different types, see the *Geometry Types* section.

- Polygon
- Mesh
- NURBS
- Bézier

CONNECTIVITY

This option is used to select the type of surface, when using a *Mesh* primitive type.

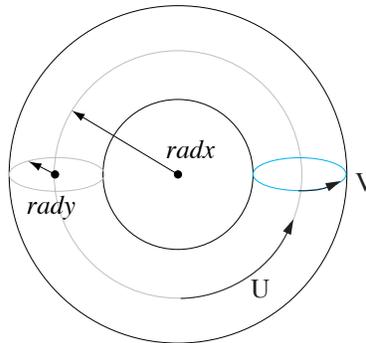
- | | |
|-------------------------|--|
| • Rows | Creates horizontal lines. |
| • Columns | Creates vertical lines. |
| • Rows & Cols | Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon). |
| • Triangles | Build the grid with Triangles. |
| • Quadrilaterals | Generates sides composed of quadrilaterals (default). |
| • Alternating Triangles | Generates triangles that are opposed; similar to the Triangles option. |

ORIENTATION

The axis along which the torus is constructed.

RADIUS */radx /rady*

The first value (*radx*) defines the radius of the torus, the second value (*rady*) determines the radius of the inner ring.



CENTER */tx /ty /tz*

Offset of torus center from object origin.

ROWS / COLUMNS */rows /cols*

The rows define the number of divisions along the torus. The columns determine the number of divisions along the torus' cross-section.

IMPERFECT

This option applies only to Bézier and NURBS types. If selected, the torus is composed of approximated nonrational curves, otherwise it is composed of perfect rational curves.

133.3 PARAMETERS – DETAIL FOLDER

U / V ORDER */orderu /orderv*

If a spline curve is selected, it is built at this order for U and V.

U ANGLE */beginangleu /endangleu*

The start and end sweep angles of the torus, if *U Wrap* is not enabled.

V ANGLE */beginanglev /endanglev*

These are the start and end angles of the cross-section circle that is swept to make the torus, if *V Wrap* is not enabled.

U / V WRAP

If U Wrap is checked, it creates a 360° cross-section. Checking V Wrap creates a torus along V by closing the primitive.

U END CAPS / V END CAPS

If *U End Caps* is checked, it puts faceted end-caps on the ends of the torus if it is less than 360°. If *V End Caps* is checked, it applies a face between the top and bottom of the torus – if the torus is open.

For more capping options, turn this parameter off, and append a Cap OP.

I33.4 USES / WORKS IN RELATION WITH

- Quickly generate torus shaped primitives
- Generate extruded squares, hexagons, etc.

I33.5 SEE ALSO

- *Cap OP* p. 468
- *Circle OP* - p. 493
- *Iso surface OP* p. 600
- *Sphere OP* p. 765
- *Tube OP* - p. 822

I34 TRACE OP

I34.1 DESCRIPTION

The Trace OP reads an image file and automatically traces it, generating a set of faces around areas exceeding a certain brightness threshold. You can control the this threshold and the resolution of the resulting faces.

I34.2 PARAMETERS – TRACE PAGE

USE DISK IMAGE

Checking this option indicates that the image file you are drawing upon is located on disk, as opposed to using a COP as a source for the image.

COP NETWORK / COP NAME

Use these pop-up menus to specify a specific COP image.

For information on COPs, see the *Composites* section of the Reference Manual.

COP FRAME */copframe*

Enter the frame value here when using a COP. Use *\$F* if you want to trace a moving image for each frame.

IMAGE INPUT

Field for entering the name of the image to be read. This field can be used to read in multiple images by entering something like:

```
$HFS/houdini/pic/butterfly5.pic
```

Tip: Enter a *\$F* in the filename to read in a sequence of animated images that are consecutively numbered. e.g. *butterfly\$F.pic*

THRESHOLD */thresh*

Brightness level value adjusts where trace outline in image occurs.

ADD POINT TEXTURE

This option allows the generation of point texture coordinates (UVs). This may occasionally be necessary when the *Convert to Poly* option is enabled.

I34.3 PARAMETERS – FILTERS PAGE

REMOVE BORDERS

When enabled, this option eliminates extraneous data along the edges of the original image so it isn't traced. This is useful for when "dirty" edges exist in the original image that you don't want traced.

BORDER WIDTH */bordwidth*

The number of pixels the removal border should be.

RESAMPLE SHAPES

Determines level of refinement (number of points) for generating trace outlines.

STEP SIZE */step*

Value controlling trace outline refinement when *Resample Shapes* is checked.

SMOOTH SHAPES

When this option is checked, the geometry is filtered to remove sharp corners.

CORNER DELTA */corner*

Value controlling corner smoothing when *Smooth Shapes* is checked.

FIT TO CURVES

If selected, the geometry at this point is converted to two-dimensional Bézier curves. Flat edges are preserved in polygons.

FITTING ERROR */error*

Value controlling accuracy of the above curve fitting process. For best results, the input should retain as many points as possible, i.e. do not select *Smooth Shapes* or *Resample Shapes*.

CONVERT TO POLY

This option will convert the above curves back into polygons.

LEVEL OF DETAIL */lod*

This value controls the accuracy of the conversion back into polygons.

HOLE FACES

This will bridge all holes in the output so that they may be rendered properly. Bézier curves and polygons can be holed, but polygonal holing sometimes produces better results. You may want to use the *Convert to Poly* option before holing.

I34.4 USES / WORKS IN RELATION WITH

Commonly followed by a *Extrude OP* to generate bevelled objects from logos, scans of geographical maps, etc. It can be animated by reading in a series of images.

I34.5 SEE ALSO

- *Extrude OP* p. 563

I35 TRAIL OP

I35.1 DESCRIPTION

The Trail OP takes an input OP and makes a trail of each point of the input OP over the past several frames, and connects the trails in different ways. It will generate trails of any input geometry, whether it is a cube translating, a deforming surface, or particles. This is useful for multi-frame ghosting effects and temporal modelling.

When using a Particle or Spring OP as input, it is important to keep the trail increment to integer values. Otherwise, the trail will not work well.

I35.2 PARAMETERS

RESULT TYPE

<i>Preserve Original</i>	Preserves the original geometry.
<i>Connect as Mesh</i>	Connects the resulting points as a mesh.
<i>Connect as Polygons</i>	Connects the resulting points as polygons.
<i>Compute Velocity</i>	As the points move faster, corresponding velocity attributes are computed. See <i>Examples</i> p. 804.

TRAIL LENGTH */length*

This sets the length of the trail by establishing the maximum number of frames for the Trail OP to use, i.e. a trail length of 25 will connect the geometry from the previous twenty-five frames.

TRAIL INCREMENT */inc*

This will skip the given number of frames to build a trail with fewer points in it, but the same length. This will lower the resolution of the trail by reducing the number of points in the trail. This is better for ghosting when using *Preserve Original*. If you set the Increment to two or more, you will see the same length trail, but fewer copies of the geometry.

CACHE SIZE */cache*

The number of frames to keep cached in available memory.

RESET CACHE

Resets the cache memory buffer.

EVALUATE WITHIN FRAME RANGE

This option specifies that the Trail OP will only evaluate, or cook, within the current frame range (\$FSTART, \$FEND). If this option is not enabled, the OP can evaluate prior to the start frame.

CONNECTIVITY

This option is used to select the type of surface, when using a *Mesh Primitive Type*.

- Rows Creates horizontal lines.
- Columns Creates vertical lines.
- Rows & Cols Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon).
- Triangles Build the grid with Triangles.
- Quadrilaterals Generates sides composed of quadrilaterals (default).
- Alternating Triangles Generates triangles that are opposed; similar to the Triangles option.

CLOSE ROWS

When selected, closes the rows in the output selection.

VELOCITY SCALE */velscale*

Scales the velocity by a specific value when Compute Velocity is selected.

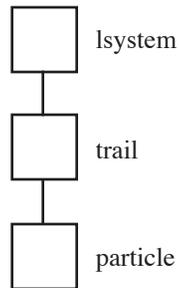
I35.3 LOCAL VARIABLES

- *none.*

135.4 EXAMPLES

VELOCITY COMPUTATION

The particles thrown off the end-most points receive a higher velocity than those close to the root of the L-system (enable Points display in Viewport Display):



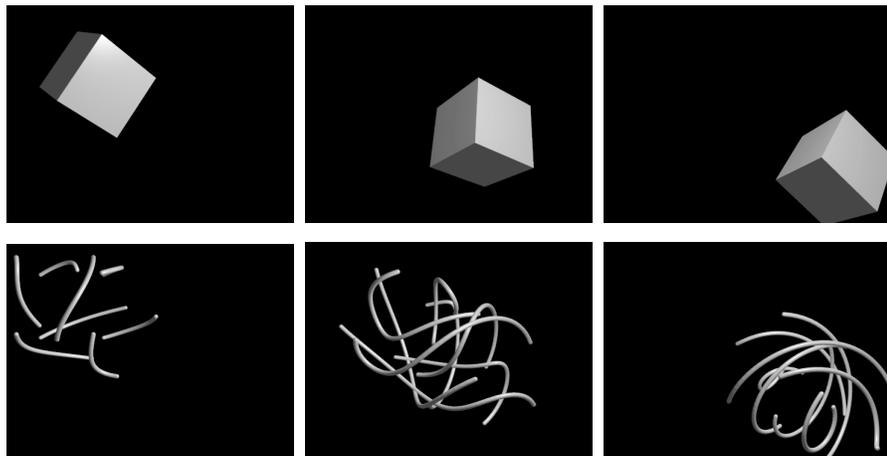
lssystem1
 /Values/Angle: $30 \cdot \sin(\$F \cdot 6) / 2$
 /Rules/Premise: A
 Rule 1: A=F+A

trail1
 Result Type: Compute Velocity

particle1
 default values – display OP

TEMPORAL MODELLING

Temporal modelling with the Trail OP – the corners of a translated and rotated cube are used as a source for the Trail OP with a *Trail Length* of 50 frames connected by *Columns*.



I36 TRANSFORM OP -

I36.1 DESCRIPTION

This OP transforms moves selected geometry in 'object space'. You can do so either explicitly via the parameters, or with the aid of the Handle* in the Viewport.

Translation changes the X Y Z positions of the points. In contrast, animating the transformation channels of an object in the Object Editor moves/scales the entire object in 'world space' and does not affect the XYZ point positions within the object geometry.

* See *Handles* p. 408 for a description of the available manipulator Handles.

I36.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

GROUP TYPE

Enter the type of elements to be referenced in the *Group* field. You can specify primitives, points, breakpoints or edges. What you enter here supplies a hint to the OP as to what kind of *Group* is being specified in order to parse it more quickly.

TRANSFORM ORDER

Sets the overall transform order for the transformations. The transform order determines the order in which transformations take place. Depending on the order, you can achieve different results using the exact same values. Choose the appropriate order from the menu.

TRANSLATE */tx /ty /tz*

These three fields move the Source geometry in the three axes.

ROTATE */rx /ry /rz*

These three fields rotate the Source geometry in the three axes.

SCALE */sx /sy /sz*

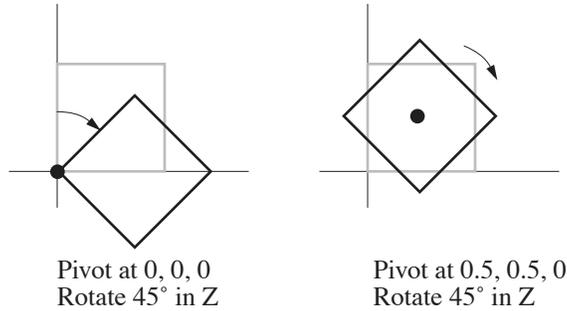
These three fields scale the input geometry in the three axes.

PIVOT

$/px /py /pz$

The pivot point for the transformations (not the same as the pivot point in the pivot channels). The pivot point edit fields allows you to define the point about which geometry scales and rotates. Altering the pivot point produces different results depending on the transformation performed on the object.

For example, during a scaling operation, if the pivot point of an object is located at: -1, -1, 0 and you wanted to scale the object by 0.5 (reduce its size by 50%) the object would scale toward the pivot point and appear to slide down and to the left.



In the example above, rotations performed on an object with different pivot points produce very different results.

Note: For veterans looking for the *Pivot Group About Centroid* – it no longer exists. Rather, previous files which use this parameter will automatically converted to using the (\$GCX, \$GCY, \$GCZ) expressions in its *Pivot* parameter.

Tip: In order to transform from the Bounding Box centre, simply enter the variables \$CEX \$CEY \$CEZ into the Pivot parameter.

UNIFORM SCALE

Uniform scale allows you to shrink or enlarge geometry along all three axes simultaneously.

RECOMPUTE POINT NORMALS

Recomputes point normals if they exist.

PRESERVE NORMAL LENGTH

When enabled, vector type attributes (i.e. normals, velocity) maintain the same length under transforms. i.e. When geometry is scaled, the normals remain constant in length.

I36.3 LOCAL VARIABLES

CEX, CEY, CEZ	Centroid of the bounding box.
XMIN, YMIN, ZMIN	The Minimum extents of the bounding box.
XMAX, YMAX, ZMAX	Maximum extents of the bounding box.
SIZEX, SIZEY, SIZEZ	Size of the bounding box in each dimension.

I36.4 MOUSE AND KEYS -

	Inclusive drag – Move both objects and Transform Jack along Construction Plane.
	Specify transformations with Value grid.
	Extend or toggle the selection (only when <i>Selection</i> mode is set to <i>Replace</i>).
	Move the jack and/or the selection up / down relative to Construction Plane, along the up vector.
	Pops-up the Operation menu.

I36.5 DESCRIPTION -

This Operation allows you to select and transform points, edges, and primitives.

SELECTING GEOMETRY

With the *Select Operation*, you select points, edges, or primitives by clicking them with the mouse, or by box-selecting them. Use the  key to extend the selection by picking more items in the default *Replace* mode (for details on selecting, see *Selecting* p. 400). Box-selecting selects all items partially caught within the bounding box.

MOVING GEOMETRY

After completing your selection, you will see a handle on the selected geometry which you can transform, rotate, scale, and otherwise work upon the selected geometry.

By default, geometry is moved along the Construction Plane. The geometry you drag will be tracked by the projection-vector cross, which provides you with feedback on the object's projection on the Construction Plane, as well as its height from the Construction Plane.

LOCKING THE SELECTION

You can lock the selection by clicking the *Secure Selection* icon. This ensures you don't accidentally lose your selection when clicking the mouse button for dragging.

MOVING UP

To drag up or down relative to the Construction Plane, hold down the **Alt** key while dragging the selection. Movement up or down is gauged by the projection vector which is grey, and the amount you have displaced it up or down indicated in red. Note that the up / down displacement starts directly above the cursor's *current* XY location, and not from where you first started dragging the selection. The up direction can be changed in the Construction Plane Operation (see *Construction Plane Operation* - p. 507), which means that **Alt** doesn't always mean straight up or down.

To move a set of selected points or primitives along their individual normals, make sure the normals are computed and displayed, then click and **Alt**-drag any normal belonging to the selected set.

Note: You may have to display the Construction Plane by clicking on the Show/Hide c-Plane icon at the bottom of the Viewport.

SCALING AND ROTATION

Translation is not the only transformation available to you in this Operation. You can also rotate the geometry about arbitrary axes, and scale it uniformly or non-uniformly along arbitrary axes, and reposition the transformation pivot. To that end, you must activate the Transform Jack described in *Handles* p. 408. For precise transformation amounts, use the parameter dialog presented in *Parameters – Transformations* Page p. 810.

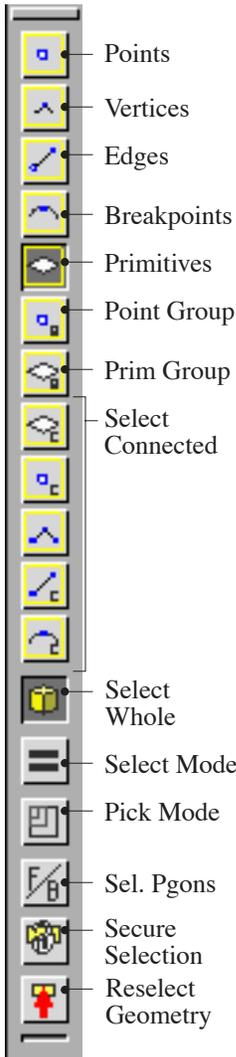
EXPLICIT TRANSFORMATIONS WITH VALUE GRID

You can explicitly define transformations by using the middle-mouse (**MB2**) to display a Value Grid.

TRANSFERRING A SELECTION TO A GROUP

Point and primitive selections can be dumped directly into a *Group* by selecting the group to you want to affect, and then clicking on one of the Add / Remove / Select / Replace icons along the top of the Viewport. Selection ranges are automatically compacted.

136.6 SELECTION ICONS -



Pause the mouse over an Icon for Heading

POINTS / VERTICES / EDGES / BREAKPOINTS / PRIMITIVES

Select one of these icons to select points, vertices, edges, breakpoints, and primitives respectively. If geometry is selected, then when switching to another selection type, it converts the selection to the new type. For example, if a primitive is selected and the type changes to points, the primitive's points will become the new selection.

POINT GROUPS / PRIMITIVE GROUPS

Allows selection of Point or Primitive Groups. The difference between the point and point group selections is that the latter selects all the points in the groups that the selected points belong to. It is similar for primitive groups.

SELECT CONNECTED POINTS / VERTICES / BREAKPOINTS

Selects connected points, vertices, or breakpoints.

SELECT WHOLE GEOMETRY

Selects the whole geometry, based on the current selection type: points, edges, breakpoints, or primitives.

SELECT MODE – ADD TOGGLE REMOVE REPLACE

These specify the action to be taken when clicking on items in the Viewport – should it add, toggle, remove, or replace items to the current selection when an item is clicked.

- When *Add* is selected, clicking an item always adds it to the selection.
- When *Toggle* is selected, clicking an item always toggles it in/out of the selection.
- When *Remove* is selected, clicking an item always removes it from the selection.
- When *Replace* is selected, clicking an item always deselects anything else that was selected, and makes the clicked item the selection. **Shift** clicking works like a toggle in *replace* mode.

PICK MODE – BOX / LASSO / BRUSH

Allows you to pick either by surrounding the selection with a *Box*, or with a *Lasso* – with which you can create arbitrary selection shapes. Using Brush Picking allows you to make your selections using Brushes.

SELECT FRONT / BACK POLYGONS 

Allows you to select polygons which are out of view. You can either select front polygons, back polygons, or both.

SECURE SELECTION

Freezes the selection so nothing more can be added or taken away from it. It prevents you from accidentally deselecting your current selection by a stray click.

If selection locking is *not* on, and you lost your selection by clicking accidentally, you can regain your selection by using *Undo* from the *Edit* menu.

■ **Tip:** You can use the  key to toggle the selection locking on and off.

RESELECT GEOMETRY

Selects the geometry previously selected over again (in case you didn't have selection locking turned on, and want to get back your selection).

I36.7 PARAMETERS – TRANSFORMATIONS PAGE**TRANSLATE**

A translation is a movement from one point in space to another. Entering XYZ values here moves the current selection by the specified amount when the *Apply Transformation* button is clicked. Translations are in the objects XYZ coordinates.

ROTATION

Entering values here rotates the current selection by the specified amount about the XYZ axes when the *Apply Transformation* button is clicked. Rotations occur around the XYZ axes of the transform jack and it's pivot. Values are in degrees.

SCALE

Entering values here scales the current selection in X, Y, and Z by the specified amount when the *Apply Transformation* button is clicked. A value of "1" leaves the selection at its original size. Values greater than 1 cause an enlargement (e.g. a value of 2 would double the selection in size along the specified axis), and values less than one shrink the selection (e.g. a value of 0.25 would shrink the selection to 25% of its original size). Scaling is done along the XYZ axes of the transform jack.

APPLY TRANSFORMATION

Clicking here applies the transformations as specified by the values in: *Translate*, *Rotation*, and *Scale*.

RESET TRANSFORMATION

Resets the transformation matrix to: Translate: 0, 0, 0; Rotation: 0, 0, 0; Scale: 1, 1, 1. Does not affect the geometry.

ABSOLUTE TRANSFORMATION

A flag that instructs *Apply Transformation* to affect the selected geometry in its current location (when off) or in its original location (when on). Thus, when the flag is off, each transformation is *relative* to the current position and orientation. When the flag is on, the transformation uses the original centroid of the selection to perform an *absolute* change of location and orientation.

PIVOT

Move the pivot (centre-point of the transform jack) to the specified XYZ location. The values are relative to world space.

HANDLE ROTATION

Enter three angles for X, Y, and Z (relative to the world XYZ axis) to determine a new orientation for the transform jack. The angles are in degrees.

136.8 PARAMETERS – CREATE SELECTION PAGE -

GEOMETRY TYPE

Select the geometry type group. The selection will only pertain to the geometry type specified. e.g. If you only wanted to group polygons.

- All Geometry – all geometry will be selected
- Bézier Curve
- Bézier Surface
- Circle
- Mesh
- Meta-ball
- NURBS Curve
- NURBS Surface
- Particles
- Polygon
- Sphere
- Tube

CREATE NEW SELECTION WITH THESE PARAMETERS...

This button creates a new selection based on the parameters specified above it.

NUMBER SUB-PAGE

Allows selection of grouping of entities by number. When checked, the options relative to this selection option are displayed. The fields available are listed below.

operation

When the Number *Enable* button is checked, this option groups entities based on a defined Pattern or by a Range.

- Group by Pattern* select a pattern in the Pattern field below.
- Group by Range* select a Range using the *Start/End* and *Select_of_* fields below.

pattern

In this field, enter the range of primitives to select. For example:

- 1-20 34 36 38 45-75
- 0-100:2 selects every other number from 0 to 100
- 0-10:2,3 selects every two of three

start/end

Activated when Operation is set to *Group by Range*. Select the start and end of the primitive/point number selection.

select _ of _

Activated when Operation is set to *Group by Range*. Select every *n*th occurrence of every *m*th entity in the above Start/End range. For example:

- enter: 1 and 2 selects 1 out of every 2 entities

NORMAL SUB-PAGE

This option is used for selecting entities based on the angle of the entity normals. When checked, the options relative to this selection option are displayed. The fields available are listed below.

The primary axis and the spread angle from the defined axis define a range of angles. If any entity normals lie within this range, then the associated entity is selected.

For example; if you want to select the polygons that are very steep in a polygon mountain terrain on the XZ axis. You would set the Direction to be 0, 1, 0 and the spread angle to around 75°. This selects all the polygons with normals that lie flat to fairly sloped. You will have grouped all of the polygons that lie flat up to polys that are at a 75° angle from the axis. You are left with all of the polygons that are 76° or greater.

direction

The default values of 0, 0, 1 create a normal vector straight up in Z, which is perpendicular to the XY plane, which becomes the primary axis. Entering something like 1, 0, 0 points the normal in +X, which is perpendicular to YZ. The plane may be

positioned at an angle by using values typed in (1, 1, 0 gives a 45° angle plane) or interactively by using the Polar jack. Values between 0 and 1 should be used.

spread angle

The value entered in this field generates an angle of deviation from the primary axis. This can be visualized as a cone where the radius of the base of the cone is defined by the Spread Angle and the axis of the cone is determined by the Direction axis. Viewing the primitive normals in the viewport, you can see that any primitives with normals that have an angle that lies in the range of angles defined by the cone will be selected and grouped.

EDGES SUB-PAGE

Allows you to group primitives by edges according to the following criteria:

edge angle

Specifies an angle between edges in which to group. Works only for primitive groups.

edge depth

Enter the depth of the edge (only for point groups).

point number

Enter the specific point numbers (only for point groups).

Note: *Edge Depth* and *Point Number* allow for a blossoming effect from the specified point. All points that are an edge depth away from the specified point are added to the group. If there are selected points already there, the *Point #* field is disabled and the blossoming occurs from the selected points.

136.9 PARAMETERS – COMBINE GROUPS PAGE -

GROUP

Group refers to the name of the group you are currently working on. You can select a new group from the ▷ menu. The available groups are filtered so that only those that match the selection type are listed. The Model Editor supports point groups and primitive groups. The group operations are ignored if the current selection type is *Edge*.

SELECTION ←- GROUP

Makes the selection reflect the contents of the group specified. Change the current group by selecting from the menu.

SELECTION ←- SELECTION + GROUP

The selection equals what is currently contained in it, plus whatever is contained in the selection group. The group remains unchanged.

SELECTION ←- SELECTION - GROUP

The selection equals what is currently contained in it, minus those components in the group that are identical to items already contained in the selection. The group remains unchanged.

SELECTION ←- SELECTION INTERSECTED WITH GROUP

Takes only those common to both the selection and the group and assigns them to the selection. The group remains unchanged.

GROUP ←- SELECTION

Makes the group equal the current selection. The selection remains unchanged.

GROUP ←- GROUP + SELECTION

Adds what is in the selection to group. The selection remains unchanged.

GROUP ←- GROUP - SELECTION

Removes from the group what it and the selection have in common. The selection remains unchanged.

ORDERED GROUPS

Single clicks select items in the order in which they are clicked on. On the other hand, box selections select items in their original creation order, unless only one entity is caught in the marquee, in which case selection ordering is preserved.

I37 TRANSFORM AXIS OP

I37.1 DESCRIPTION

Transform Axis transforms the input geometry with respect to a specified axis. This is useful to derive an animatable channel to transform an object about a single axis.

I37.2 PARAMETERS

GROUP

Subset of geometry to transform.

GROUP TYPE

The type of elements referenced in the Group field.

ORIGIN */orig*

The pivot to do the transform with.

DIRECTION */dir*

The orientation of the axis to transform with.

TRANSLATE */trans*

Amount of translation along the axis.

ROTATE */rot*

Amount of rotation about the axis.

SCALE */scale*

Non-uniform scaling along the axis.

RECOMPUTE POINT NORMALS

Recomputes point normals if they exist.

PRESERVE NORMAL LENGTH

Normal lengths remain unaffected.

137.3 LOCAL VARIABLES

CEX, CEY, CEZ	The centroid of the input.
GCX, GCY, GCZ	The centroid of the input group.
XMIN, XMAX	The X extents of the bounding box of the input.
YMIN, YMAX	The Y extents of the bounding box of the input.
ZMIN, ZMAX	The Z extents of the bounding box of the input.
SIZEX, SIZEY, SIZEZ	The size of the bounding box of the input.

137.4 SEE ALSO

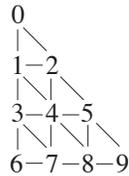
- *Primitive OP* p. 697
- *Point OP* p. 667
- *Transform OP* - p. 805

I38 TRI-BÉZIER OP

I38.1 DESCRIPTION

The Tri-Bezier operation creates a triangular Bezier surface using a user specified group of points as the control points of the Bezier. The order of the points is important for creating the surface. The number of points should be $(order+1)*order/2$. For example: order 4 requires 10 points, order 3 – 6 points, and order 2 (a triangle) 3 points. Extra points are ignored.

The order of the points should be as follows for an order 4 patch:



I38.2 PARAMETERS

GROUP

This is the points, in order, to be used.

ORDER */order*

This is the order of the resulting patch.

I38.3 SEE ALSO

- *Convert OP* p. 512

I39 TRI-STRIP OP

I39.1 DESCRIPTION

The Tri-Strip operation converts polygons to triangle strips. Triangle strips are faster to display and require less memory than the equivalent triangles. However, most editing functions require geometry to be in the form of triangles rather than triangle strips. You can convert back to triangles using a Convert operation.

Note that if you are sending in geometry which is changing in topology, the Tri-Strip operation may triangulate polygons differently in different frames, causing popping if the polygons weren't planar. If this is a problem, triangulate non-planar polygons first using a Divide operation.

I39.2 PARAMETERS

SOURCE GROUP

The polygons that should be tristripped

CONSTRAIN STRIP LENGTH

Should the maximum length of strips be bounded?

MAX STRIP LENGTH */maxstriplength*

The maximum number of triangles in any one triangle strip.

I39.3 SEE ALSO

- *Convert OP* p. 512

I40 TRIM OP

I40.1 DESCRIPTION

With the Trim OP you can cut out parts of a spline surface, or uncut previously cut pieces. When a portion of the surface is trimmed, it is not actually removed from the surface; instead, that part is made invisible. This means that you can still modify the surface (modify the position of its points, for instance) that is not displayed in order to affect the part that is displayed.

The surface can be trimmed by specifying open or closed profiles as inside or outside regions. The profiles need not be contained within the domain (UV space) of the surface; they can also be nested.

Open profiles are treated as follows: if both ends of the profile are inside the surface, the ends are connected to one another; if the profile's ends are outside the domain of the surface they are projected onto, that part of the surface appears to be cut away.

You will usually need a Trim, Bridge, or Profile OP after a Project OP.

- Use a Trim OP to cut a hole in the projected surface.
- Use a Bridge OP to skin the profile curve to another profile curve.
- Use a Profile OP to extract the curve on surface or remap it's position.

selection method – winding rule

The selection method employed for clarifying overlapping trim loops is the *winding rule*, which executes overlapping commands instead of having them cancel each other out.

Tip: Since only surfaces containing profile curves can be trimmed, you will always need a Project or Carve OP in the chain above the Trim OP.

I40.2 PARAMETERS

GROUP

This field allows you to specify the group that you would like to trim. You can select the group from the pop-up menu, or specify a points and primitives range.

You can specify profile curves within the group by providing a profile pattern (e.g. *.3 specifies the fourth profile in all spline surfaces).

KEEP

Keep Outside

Trims the interior of the curve, or makes a hole in the display of the surface. Keeping what's outside the profile curve generates an outer trim loop in order to define the limits of the surface to be displayed—the surface that's outside of the profile curve.

<i>Keep Inside</i>	Keeps the interior of the curve and removes the display of everything else.
<i>Keep Natural</i>	Trim based on the natural orientation of the profiles, be they open or not. Counter-clockwise profiles keep their interior, generating a result similar to <i>Keep Inside</i> . Clockwise profiles discard their interior, similar to <i>Keep Outside</i> , and may require an explicit outer trim-loop if none is present.
<i>Untrim</i>	Turns the trim curve into a plain profile.

PROCESS PROFILES INDIVIDUALLY

When this option is off, the trim loops in the group (or all the loops on the surfaces if no group has been specified) will be considered together to form a region. It will report the first region that is found. That is, if more than one closed loop could be formed by joining the lines, there is no guarantee that the region is trimmed. Also, if there is a closed loop in the group of loops, then just that loop is used.

If the loops on the surface don't form a closed loop, then the OP will attempt to form a region by using the boundary of the region. If all the loops make a total of two intersections with the boundary, then it will attempt to form the loop by forming it around the boundary.

example

Use the Carve OP to extract four profiles: two in U, and two in V. Pipe that into a Trim OP and turn this option off. The four profiles will define a region to be trimmed. Notice that the profile end-points do not coincide, and the profiles are not parametrically continuous, nor are they created in the proper order. Despite all this, the Trim OP is able to figure out the hole.

BUILD OUTER TRIM LOOP EXPLICITLY

This option allows you to specify that an outer trim loop be built. It is useful where you have more than one profile curve on the surface and are performing several successive trim operations involving both the *Keep Inside* and *Keep Outside* options (see example, below).

Tip: An outer trim loop must be generated the first time you punch a hole in the surface, but not if you just keep the contents of that hole and throw away the rest. By default, the outer loop (which goes all around the domain boundary) is built for you automatically. Sometimes, however, you first do a “Keep Inside”, then a “Keep Outside” with an area that's not inside the preserved regions, so you may want the outer curve at that point. That is when this parameter is useful.

TRIMMING TOLERANCE */trimtol*

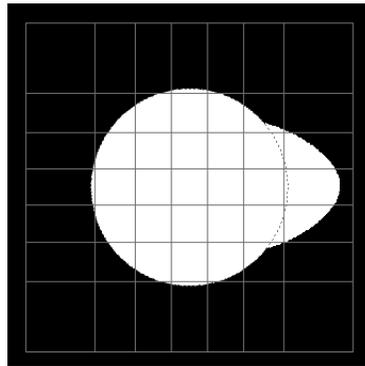
How close two trim curves must be to each other or to the edge of the patch in order to be considered an intersection.

ALTITUDE

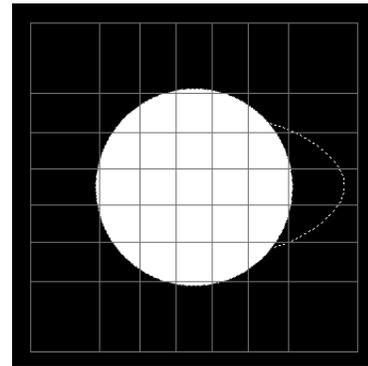
You can specify the altitude of the trim. The \$ALTITUDE variable is the surface's current altitude. This marks the transition for the surface from trimmed in to trimmed out.

140.3 EXAMPLE

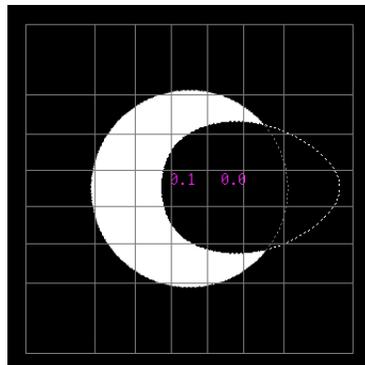
The following results were obtained by using a Project OP to project two NURBS circles onto a NURBS grid. Then two Trim OPs were added, one after the other, to the Project OP. The first Trim OP was set to *Keep Inside*, while the second Trim OP had its operation changed as indicated.



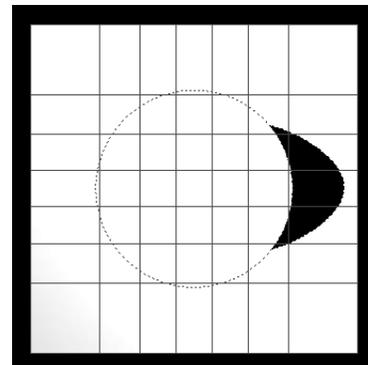
Keep Inside



Untrim



Keep Outside
(Build Outer Loop: Off)



Keep Outside
(Build Outer Loop: On)

The illustrations show a Gouraud shaded view of the resulting geometry.

I41 TUBE OP -

I41.1 DESCRIPTION

Generates open or closed tubes, cones, or pyramids along the X, Y or Z axes.

I41.2 PARAMETERS – TUBE PAGE

PRIMITIVE TYPE

Select from the following types. For information on the different types, see the *Geometry Types* section.

- Primitive
- Polygon
- Mesh
- NURBS
- Bézier

CONNECTIVITY

This option is used to select the type of surface, when using a *Mesh Primitive Type*.

- Rows Creates horizontal lines.
- Columns Creates vertical lines.
- Rows & Cols Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon).
- Triangles Build the grid with Triangles.
- Quadrilaterals Generates sides composed of quadrilaterals (default).
- Alternating Triangles Generates triangles that are opposed; similar to the Triangles option.

ORIENTATION

Primary axis of tube (long axis).

CENTER */tx /ty /tz*

Location of the tube center from the object origin.

RADIUS */rad1 /rad2*

The first field is the radius of the top of the tube and the second field represents the radius of the bottom of the tube.

HEIGHT */height*

The height of the tube.

141.3 PARAMETERS – DETAIL PAGE

ROWS / COLUMNS */rows /cols*

Number of rows / columns in tube.

U / V ORDER

If a spline surface is selected, it is built at this order for U and V.

IMPERFECT

This option applies only to Bézier and NURBS types. If selected, the tube is an approximated nonrational curve, otherwise it is a perfect rational curve.

END CAPS

If selected, it adds faceted end caps to the ends of the tube.

141.4 USES / WORKS IN RELATION WITH

Generates tubes from hollow tubes to cones and pyramids.

141.5 MOUSE AND KEYS -

none Volatile key

 Pops-up the Operation menu.

141.6 DESCRIPTION -

This Operation is used to create tubes and cones. Click and drag the mouse on the Construction Plane to generate a tube whose radii and height are specified by the drag.

Create cones by entering a value of zero for either the top or bottom radius in the Parameters.

Clicking the mouse button without dragging places a tube with radii and height as specified in the Parameters dialog box (default of 1) at the location of the mouse click. The radii of the tube are aligned with the X and Y axes of the Construction Plane.

Typing **Enter** places a tube or cone whose size and position are specified in the Parameters dialog.

If an odd aspect ratio was previously entered in the Parameters dialog, clicking and dragging will produce tubes which maintain that aspect ratio. This can be reset by clicking on the *Reset Radii* button.

141.7 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog for this Operation. See *Parameters – Tube Page* - p. 824.

PRIMITIVE / POLYGON / MESH / NURBS / BÉZIER

Click these buttons to select which type of tube you want to draw. For differences between the five types, see *Geometry Types*.

RESET RADII

Changes the radii and height of the tube in the parameters dialog back to 1.

141.8 TUBE OPERATION MENU -

Call up the menu for this Operation by using **Ctrl** .

PRIM / POLY / MESH / NURBS / BÉZIER TUBE / / / /

These five menu items allow you to specify whether you want to create a tube as a primitive, polygonal sets, mesh, NURBS or Bézier surface.

BUILD DEFAULT TUBE **Enter**

Constructs a default tube in the viewport. The tube is built as currently specified in the Parameters dialog, with its radii aligned with the X and Y axes of the Construction Plane.

141.9 PARAMETERS – TUBE PAGE -

CONNECTIVITY

- Rows Creates horizontal lines.
- Columns Creates vertical lines.
- Rows & Cols Both Rows and Columns. Looks like Quads in wire frame display, but all polygons are open (if the primitive type is polygon).

- Triangles Build the grid with Triangles.
- Quadrilaterals Generates sides composed of quadrilaterals (default).
- Alternating Triangles Generates triangles that are opposed; similar to the Triangles option.

TOP / BOTTOM RADIUS

The radius of the top and bottom of the tube. Making one of these radii zero produces a cone. This is the size of the tube created if you click on the Construction Plane without dragging. If you click and drag, the radii vary with the amount of drag, with the top / bottom ratio preserved.

HEIGHT

The height of the tube. If you click and drag on the Construction Plane, the height varies with the amount of drag, and is proportional to the radii / height ratio.

CENTER

Determines the location of the center of the tube. This value is updated whenever you click (and drag) to create a tube. A new tube will be positioned here if you type Enter.

ROWS / COLUMNS

The number of rows and columns of a mesh or imperfect NURBS / Bézier tube. The more columns, the rounder the tube. Unless you are planning to bend the tube along its height, keep the number of rows low (e.g. 2). A NURBS or Bézier tube should have at least V order rows and U order-1 columns (it is wrapped in U).

U ORDER / V ORDER

Sets the U and V spline order of the NURBS or Bézier surface when building a tube of one of these two types. The lowest order is 2 (linear); the highest is 11. Cubic tubes are built by default.

IMPERFECT

Specifies whether the NURBS / Bézier tube should be built using rational or non-rational splines. A perfect tube has a rational topology – one that associates non-unit weights with (some) of its vertices. Furthermore, a perfect tube has a predefined number and positions of CVs for any given spline order. An imperfect tube is non-rational and its number of CVs isn't that strictly determined by its order.

Rational tubes built this way yield a mathematically perfect shape; however, given their special definition, perfect tubes are not always the ideal choice for further modeling of their points. Besides, they represent heavier geometry and may put more pressure both on the CPU and RAM. In practice, you will find imperfect tube to be a better modelling choice, so it is advisable to build perfect tubes only when perfect shapes are paramount.

141.10 PARAMETERS – CAPS PAGE -

Normally, a tube has a hole at either end. Adding caps covers up these holes.

FIRST / LAST U CAP

Select an option from the menu:

<i>No End Cap</i>	Leave the side of the primitive unaffected.
<i>End Cap Faceted</i>	Cap by creating a face a similar type which uses unique points.
<i>End Cap Shared</i>	Cap by creating a face a similar type which shares points with the original primitive.
<i>End Cap Rounded</i>	Cap by creating / extending a mesh containing a number of concentric circular regions. this creates a bullet or domed shaped cap.

DIVISIONS

Determines the number of cross-sections the end cap will have. Works for all types except primitive tube (it would no longer be a primitive tube if it had end caps).

SCALE

Affects the height of the rounded cap (both positive and negative).

I42 TWIST OP

I42.1 DESCRIPTION

Generalized non-linear deformations such as bend, linear taper, shear, squash and stretch, taper and twist. Each one will distort the object in one or more axes to variable degrees.

I42.2 PARAMETERS

All deformations occur around the primary axis. If the primary axis is Y, then the Y value does not get changed. For deformations that use the secondary axis, the points along the secondary axis will not be changed. Edges that are parallel to the primary axis will remain parallel during the transformation.

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

OPERATION

This menu allows you to select a type of non-linear deformation. Select from the following options:

<i>Twist</i>	Rotates the input geometry around the primary axis.
<i>Bend</i>	Bends the input geometry about the primary axis while keeping points on the secondary axis stationary.
<i>Shear</i>	Shears the input geometry along the secondary axis while looking down the primary axis.
<i>Taper</i>	Tapers the input geometry along the secondary axis while looking down the primary axis.
<i>Linear Taper</i>	Tapers the input geometry as with the <i>Taper</i> option; however, only the edges remain straight through the taper operation.
<i>Squash and Stretch</i>	Traditional animator's bounce tools.

PRIMARY AND SECONDARY AXIS

These menus allows you to select the primary and secondary axes for the deformation. The selected deformation will first occur in the primary axis and then the secondary axis.

PIVOT */px /py /pz*

This field allows you to choose the origin of the deformation.

STRENGTH / ROLL OFF */strength /roll*

The strength of the effect being applied. The Rolloff determines an accentuation of the effect being applied. When you are using different types of transformations this strength / roll will have different effects:

<i>Bend</i>	Strength and Roll are used to control the extremities of the geometry (try a value of 0.5).
<i>Twist</i>	Strength and Roll are used to affect the twist amount based on the distance.
<i>Shear</i>	Strength and Roll are used to affect the shear amount based on distance.
<i>Taper</i>	Strength and Roll are used to affect the direction of the bow (inwards vs. outwards).
<i>Linear Taper</i>	Strength and Roll have no effect for this option.
<i>Squash and Stretch</i>	Strength and Roll are used to maintain the apparent volume of the source geometry.

Typically, *Rolloff* should equal 1 – which spreads the effect evenly (although not being limited to) across the bounds of the geometry. Values higher than 1 iterate the effect multiple times through the same range. If *Rolloff* equals 0, then the effect may be localised to a small segment at the centre of the deformed geometry and *Strength* may not appear to work properly.

Note: To be certain to see the effects of the Twist OP, make sure you have enough divisions along the edges. By using a centre that is different from that of the object you can improve your control of the object. Try moving the pivot point to the bottom of an object that you are squashing and stretching.

142.3 INPUTS / GEOMETRY TYPES

INPUT 1 – TWIST SOURCE

Can be any geometry, including NURBS and Béziers), but should have sufficient points for a better deformation. Use the Divide or Refine OPs to achieve this.

I42.4 LOCAL VARIABLES

<i>CEX, CEY, CEZ</i>	Defines the centroid of the input geometry
<i>XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX</i>	Defines the extent of the bounding box for the input geometry
<i>SIZEX, SIZEY, SIZEZ</i>	The size of the bounding box of the input geometry

I42.5 USES / WORKS IN RELATION WITH

Generalized non-linear deformations such as bend, linear taper, shear, squash and stretch, taper and twist.

I42.6 SEE ALSO

- *Divide OP* p. 549
- *Refine OP* p. 719

I42.7 EXAMPLE



You can twist any geometry.

I43 UNIX OP

I43.1 DESCRIPTION

The Unix OP allows direct access to command-line based UNIX programs that perform a host of geometry functions (see the *Stand Alone Tools* section).

I43.2 PARAMETERS

UNIX COMMAND

Enter a UNIX command string here. The geometry input to this OP will be sent to the standard input of the UNIX command specified. For geometry tools such as the *g*-tools provided with Houdini, an input argument of “stdin” will indicate that they should take their input from the standard input of the process. The program should send its output to its standard output in a readable geometry format such as *.geo* or *.bgeo*.

OUTPUT FORMAT

Select *ASCII* to have Houdini *.geo* format sent to the command or “Binary” to have Houdini *.bgeo* format sent to it instead.

I43.3 USES / WORKS IN RELATION WITH

Gives access to a number of Houdini or UNIX tools in *\$HFS/bin* usually preceded by a *g*. (*g* tools support all geo types).

I43.4 SEE ALSO

- *Standalone Tools* section

I44 UNPASTE OP

I44.1 DESCRIPTION

The Unpaste OP removes one or more pasted surfaces from a paste hierarchy, causing the hierarchy to update. It can keep either the unpasted surfaces or what remains pasted in the hierarchy after the removal of the unpasted surfaces.

The OP accepts both pasted surfaces and entire paste hierarchies as inputs. For example, to turn a paste hierarchy into a set of unrelated surfaces, enter the hierarchy number in the *Group* field.

By preserving the hierarchical structure of the unpasted surfaces, the resulting sub-hierarchy can be reapplied properly to another hierarchy later on.

The shape and the size of the sub-hierarchy may change considerably as a result of the unpasting operation. This is because the feature surfaces are always mapped onto the domain of the base surface, and the size of the domain is completely unrelated to the size of the actual surface.

By unpasting the root of the paste hierarchy the whole hierarchy becomes undone.

UNPASTED SURFACE HIERARCHIES

An unpasted surface or sub-hierarchy can be repasted later with the Paste OP. One case of repasting is worth mentioning for its practical use.

Assume you have used the Paste OP to spawn a new surface as a detail added to the base surface. Now you are ready to model the new feature. You can do so by working on the pasted feature, making sure to affect only its points and not the points of the base surface as well. If the point density of the model is high, it may be more convenient to model the feature separately, then re-attach it to the paste hierarchy as if it had never been removed.

There are three easy steps to achieve this goal:

- Use the Unpaste OP with the default parameters: keep the unpasted part and preserve the shape of the feature together with its hierarchical information. Make sure to specify the feature's primitive number in the Group field. Notice the shape of the feature has not changed.
- Model the stand-alone feature with Houdini's vast array of tools.
- Repaste the feature on the same base surface using the "As Is" tab with shape preservation enabled.

I44.2 PARAMETERS

GROUP

By specifying a group, you can act on a subset of pasted surfaces or paste hierarchies. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

INCLUDE ALL NESTED CHILDREN

Unpaste the surface with all its nested nodes.

KEEP UNPASTED PART

When enabled, it keeps only the unpasted surface(s).

preserve pasted shape

Preserve the shape they had when pasted.

preserve hierarchical structure

Extract as a new paste hierarchy.

KEEP REMAINING HIERARCHY

Enabling this button deletes the unpasted surface(s).

I44.3 SEE ALSO

- *Delete OP* p. 543
- *Paste OP* p. 661

I45 UV EDIT OP

I45.1 DESCRIPTION

UV Edit allows interactive editing of the UVs on the source geometry. Multiple edits can be made within a single UV Edit operation.

The UV Edit operation will automatically detect whether the input geometry uses point or vertex UV attributes, and will affect these attributes appropriately.

3D transforms are allowed in the UV Edit operation, as the UV attribute is actually 3 coordinates: U, V, and W. When used interactively, translates and scales are along the U/V axes while rotations are about the W axis.

The UV Edit operation is cumulative, meaning that several operations can be performed within a single node in the network. UV Edit is not animatable. To animate UV transformations, use UV Transform or other UV deformers.

I45.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

GROUP TYPE

Enter the type of elements to be referenced in the *Group* field. You can specify primitives, points, breakpoints or edges. What you enter here supplies a hint to the OP as to what kind of *Group* is being specified in order to parse it more quickly.

TRANSFORM ORDER

Sets the overall transform order for the transformations. The transform order determines the order in which transformations take place. Depending on the order, you can achieve different results using the exact same values.

TRANSLATE */tx /ty /tz*

These three fields move the Source geometry in the three axes.

ROTATE */rx /ry /rz*

These three fields rotate the Source geometry in the three axes.

SCALE */sx /sy /sz*

These three fields scale the input geometry in the three axes.

PIVOT */px /py /pz*

The pivot point for the transformations (not the same as the pivot point in the pivot channels). The pivot point edit fields allows you to define the point about which geometry scales and rotates. Altering the pivot point produces different results depending on the transformation performed on the object.

PIVOT ABOUT GROUP CENTROID

Set the pivot to the center of the input group.

HANDLE FOLLOWS GROUP

Move the handle to the center of the group when a new operation begins.

HANDLE PIVOT

The handle pivot point.

HANDLE ROTATION

The handle rotation.

RESET ON EACH NEW OPERATION

If the handle is detached during an operation, restore it when a new operation begins.

SOFT RADIUS */rad*

The area of influence.

SOFT TYPE

Type of rolloff function.

TANGENT ANGLES */tandeg*

Angles of the cubic rolloff function's tangents. The first value applies to the tangent farthest from the source point, the second applies to the tangent closest to the source point.

KERNEL FUNCTION

Metaball kernel to use when rolloff Type is *Metaball*.

METRIC

Measure of distance to use when performing a soft transform. XYZ corresponds to distance on the geometry. UVW corresponds to distance in UVW (texture) space. UV ignores the third dimension in texture space.

IGNORE POINT CONNECTIVITY

Affect only points connected to the points in the group, or all points within the radius.

IGNORE UV CONNECTIVITY

Affect only vertices connected to the vertices in the group as seen in the UV viewport.

START NEW UV TRANSFORM

Begin a new UV Transform. Keep all changes, but reset the operation parameters.

RESET ALL CHANGES

Discard all changes made while using this SOP, and reset the operation parameters.

I45.3 SEE ALSO

- *UV Transform OP* p. 841
- *UV Fuse OP* p. 836

I46 UV FUSE OP

I46.1 DESCRIPTION

The UV Fuse operation allows you to fuse UV attributes on the source geometry.

UV Fuse can fuse UV attributes based on their proximity to each other, and can position the fused UVs using a variety of methods. The effects of the UV Fuse operation are most visible when using the UV viewport.

One possible use for UV Fuse is to re-connect UV attributes along boundaries that were torn apart by the UV Unwrap operation.

I46.2 PARAMETERS

GROUP

Subset of geometry with UV attributes to fuse.

GROUP TYPE

The type of elements referenced in the Group field.

DISTANCE TAB

Positioning Method *Average* assigns an average UV coordinate to groups of fused UV attributes. *First in Group* assigns the first UV coordinate to all other fused UV attributes.

Distance (/dist) Threshold distance for consolidation.

Metric This is the measure of distance to use when fusing UV attributes. *XYZ* corresponds to the distance along the surface of the geometry; *UVW* corresponds to distance in UVW (texture) space; and *UV* ignores the third dimension (W) in texture space.

MANUAL TAB

Set all UV attributes to a specific value.

Texture Coordinate The value to assign to UV attributes when Manual is selected.

GRID TAB

Grid Type (/gridtype) How to specify the grid size.

Grid Spacing (/gridspacing) The number of units between each grid line.

Grid Lines (/gridlines) The number of grid lines every unit.

Grid Power 2 (/gridpow2) The same as Grid Lines, but a power of 2 is specified (i.e. $2^1 = 2$, $2^2=4$, $2^3=8$... $2^7=128$.. $2^9=512$)

Grid Offset A number between 0 .0 and 1.0 which specifies what offset the grid should have from an origin of (0, 0, 0).

I46.3 SEE ALSO

- *UV Unwrap OP* p. 850
- *UV Transform OP* p. 841
- *UV Edit OP* p. 833

I47 UV PROJECT

I47.1 DESCRIPTION

UV Project assigns texture UV coordinates to the Source geometry for use in texture and bump mapping. UVs are projected onto the geometry via a transformed projection geometry.

The effect of this operation is best visualized in the UV viewport, or with textures turned on in the 3D viewport. Then you can use the interactive handles to position an incoming texture using this SOP.

UV Project creates the UV texture attribute if it does not already exist. The attribute class (Vertices or Points) is determined by the Group Type. It is recommended that UVs be applied to vertices, since this allows fine control on polygonal geometry and the ability to fix seams at the boundary of a texture.

Note: For closed mesh Bezier & NURBS surfaces, projections with boundaries will result in seams. UV Texture can be used to open these surfaces automatically. Alternatively, convert the surface to polygons using a Convert operation prior to applying UV Project.

I47.2 PARAMETERS

GROUP

Subset of geometry to apply texture UV coordinates to.

GROUP TYPE

The type of elements referenced in the Group field, and the class of UV texture attribute to use.

PROJECTION

Type of projection geometry to use.

INNER RADIUS */torrad*

Inner radius of the torus used in a Toroidal projection. The outer radius is always 0.5.

TRANSFORM ORDER

Order in which transformations occur.

ROTATE ORDER

Order in which rotations occur.

TRANSLATE */tx /ty /tz*

These three fields move the Source geometry in the three axes.

ROTATE */rx /ry /rz*

These three fields rotate the Source geometry in the three axes.

SCALE */sx /sy /sz*

These three fields scale the input geometry in the three axes.

PIVOT */px /py /pz*

The pivot point for the transformations (not the same as the pivot point in the pivot channels). The pivot point edit fields allows you to define the point about which geometry scales and rotates. Altering the pivot point produces different results depending on the transformation performed on the object.

INITIALIZE TRANSFORMATION

Automatically fit the projection geometry to the group's bounding box.

U RANGE */urange*

The location of the left and right edges, respectively, of the texture on the projection geometry.

V RANGE */vrange*

The location of the bottom and top edges, respectively, of the texture on the projection geometry.

ANGLE */angle*

Rotates texture coordinates about the point (0.5, 0.5) in UV texture space.

FIX BOUNDARY SEAMS

Makes sure the texture wraps around correctly.
This only works for vertex UV attributes.

FIX POLES

Use the UV coordinates of neighbouring vertices to improve undefined projections at the poles of the projection geometry. (For example, the north and south poles of the Polar projection).

POLE RADIUS */polerad*

The distance from the exact pole within which a vertex is treated as being in that pole.

147.3 SEE ALSO

- *UV Texture OP* p. 843
- *UV Unwrap OP* p. 850

I48 UV TRANSFORM OP

I48.1 DESCRIPTION

The UV Transform operation transforms UV texture coordinates on the source geometry. Unlike the UV Edit operation, UV Transform supports animation of UVs.

UV Transform will automatically detect whether the input geometry uses point or vertex UV attributes, and will affect these attributes appropriately.

3D transforms are allowed in the UV Transform operation, as the UV attribute is actually 3 coordinates: U, V, and W. When used interactively, translates and scales are along the U/V axes while rotations are about the W axis.

I48.2 PARAMETERS

GROUP

Subset of geometry to apply current operation to.

GROUP TYPE

The type of elements referenced in the Group field.

TRANSFORM ORDER

Order in which transformations occur.

ROTATE ORDER

Order in which rotations occur.

TRANSLATE */tx /ty /tz*

These three fields move the Source geometry in the three axes.

ROTATE */rx /ry /rz*

These three fields rotate the Source geometry in the three axes.

SCALE */sx /sy /sz*

These three fields scale the input geometry in the three axes.

PIVOT */px /py /pz*

The pivot point for the transformations (not the same as the pivot point in the pivot channels). The pivot point edit fields allows you to define the point about which

geometry scales and rotates. Altering the pivot point produces different results depending on the transformation performed on the object.

SOFT RADIUS */rad*

The area of influence.

SOFT TYPE

Type of rolloff function.

TANGENT ANGLES */tandeg*

Angles of the cubic rolloff function's tangents. The first value applies to the tangent farthest from the source point, the second applies to the tangent closest to the source point.

KERNEL FUNCTION

Metaball kernel to use when rolloff Type is "Meta-ball"

METRIC

Measure of distance to use when performing a soft transform. XYZ corresponds to distance on the geometry. UVW corresponds to distance in UVW (texture) space. UV ignores the third dimension in texture space.

IGNORE POINT CONNECTIVITY

Affect only points connected to the points in the group, or all points within the radius.

IGNORE UV CONNECTIVITY

Affect only vertices connected to the vertices in the group as seen in the UV Viewport.

148.3 SEE ALSO

- *UV Edit OP* p. 833
- *UV Fuse OP* p. 836

I49 UV TEXTURE OP

I49.1 DESCRIPTION

The UV Texture OP assigns texture UV coordinates to the source geometry for use in texture and bump mapping.

When using one of the spline-based methods, specifying a paste hierarchy in the group field will propagate the computation of texture coordinates to all of its nodes. Projection methods will typically yield smoother texture continuity between pasted surfaces than any of the spline methods. Sometimes it helps ensuring that pasted features are chord-length parameterized with the Basis operation.

Note: When the projection type is cylindrical or polar, closed mesh Bezier & NURBS surfaces will be opened. At least one row/column of vertices will be added (possibly more for NURBS). This is to prevent poor interpolation of texture coordinates at the seam of the join.

FIXING SEAMS & UNROLLING GEOMETRY

When to fix seams and when to unroll the wrapped geometry?
(the following discussing pertains only to face and hull primitives).

If a texture type requires fixing of seams and the texture is applied to vertices, the wrapped primitives are unrolled *before* computing the texture coordinates. Unrolling a wrapped primitive turns it into an open primitive whose new vertices use the same points as the vertices they have been unrolled from. Thus, unrolling does not change the point count, nor does it allow cracks to appear further down the road. Explicit unrolling, using the Primitive OP, is not required.

The seam fixing is done *after* computing the texture coordinates. It is required whether textures are applied to vertices or points, and it's done in u, v, or both.

The following texture types require fixing of seams:

- Cylindrical (seams fixed in u)
- Polar (seams fixed in u and v)
- Row/col (seams fixed in u and v)
- Spline types: uniform, chord length, and average (seams fixed in u and v)

Note: When the projection type is cylindrical or polar, closed meshes, Bézier & NURBS surfaces will be opened. At least one row/column of vertices will be added (possibly more for NURBS). This is to prevent poor interpolation of texture coordinates at the seam of the join.

POLAR MAPPING RENDERMAN TEXTURES

To get a Texture OP to correctly render Renderman textures that are polar mapped to a sphere, you should use a Polar TOP, because using a Texture OP with polar mapping will not provide correct results.

The reason for this is because at the pole of the sphere, all the points are directly on the (local) Y axis, so the radial component can't really be set correctly (just try to compute: $\text{atan}(0, 0)$!).

Using a Polar TOP works, because during the interpolation of coordinates assigned to points on geometry, every pixel's texture coordinates are computed, so it doesn't know about interpolation of attributes.

Another solution is to use a Carve OP to carve off the very smallest portion off the top and bottom of the sphere. Values something like 0.005 and 0.995 are sufficient. Keep both the Inside and Outside. In some cases, Consolidating Points after the texture might also work.

149.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

TEXTURE TYPE

The *Face*, *Uniform Spline*, and *Arc-Length Spline* texturing methods accept spline curves as well as polygons.

When using one of the spline-based methods, specifying a paste hierarchy in the *Group* field will propagate the computation of texture coordinates to all of its nodes. Projection methods will typically yield smoother texture continuity between pasted surfaces than any of the spline methods. Sometimes it helps ensuring that pasted features are Chord-length parameterized with the Basis OP.

<i>Orthographic</i>	Direct projection from axis.
<i>Polar</i>	Wrap spherically in axis direction.
<i>Cylindrical</i>	Wrap cylindrically in axis direction.
<i>Rows and Columns</i>	For geometry constructed as a mesh (Grid, Sphere, Tube, Skin, and Sweep). The U coordinates are placed along rows, and the V coordinates along columns. This is good for texturing curved meshes such as car fenders where you cannot project from any one axis.

<i>Face</i>	Maps a copy of the texture onto every face. You should make points unique using a Facet OP, before using this function in the Texture OP. The map is graphically projected to each face along its normal, so the texture is oriented properly for each face. However, the map is not scaled to fit each polygon, nor is it distorted by the shape of each polygon. If the geometry changes in size in object space, the texture does not “stick” to the geometry. It is best suited to texturing objects that represent chunks of rock and brick, as the textures will likely not match at the edges between polygons.
<i>Modify Source</i>	If the Source already has texture UV coordinates, they are maintained. You can offset and scale them, however, using <i>Scale</i> and <i>Offset</i> .
<i>Uniform Spline</i>	This projection type operates only on NURBS and Bézier surfaces. It samples the domain space (i.e. the basis) of each surface uniformly in U and V and assigns those (u,v) values as texture coordinates to the surface points or vertices. To ensure continuity between the texture space of adjacent surfaces insert a Basis OP before the Texture OP and toggle “Concatenate” on to merge the spline bases in U and/or V.
<i>Average Spline</i>	Stores the average of ‘degree’ successive knots into the texture attribute. These averages are known as “Greville points”. This method and “Uniform Spline” are recommended for pasted surfaces.
<i>Arc Length Spline</i>	This method is similar to the “Uniform Spline” method since it relies on the underlying spline basis when computing the texture coordinates. Both methods generate texture coordinates in the same range, bounded by the minimum and maximum knot values. The difference between the two spline methods lies in the spacing between successive texture coordinates. The uniform method samples the parameter space uniformly. The Arc-Length method chooses the texture coordinates based on surface arc-lengths.
<i>Perspective From Camera</i>	The texture coordinates are assigned so that the world space of the object can be textured to fit the projection of the camera exactly. If any points are behind the near clipping plane or beyond the far clipping plane, the texture coordinates (0, 0, 0) are assigned.

Since the *Uniform Spline* method relies heavily on the parametric fabric of the surface, the resulting texture will tend to squash and stretch given uneven surface parameterizations. The *Arc-Length* method reduces this effect by relating the texture space directly to the world space of the surface, while ensuring that the size and origin of the generated texture space coincide with those of the underlying domain.

CAMERA NAME

This is used when the *Perspective From Camera* texture type is selected. The menu is used to select which light or camera to project the perspective coordinates from.

PROJECTION AXIS

X, Y or Z direction.

APPLY TO

When *Natural* location is selected, the UV's will be applied to the vertices when using the following mapping types:

- Polar
- Cylindrical
- Rows and Columns
- Face

Orthographic, Uniform Spline, Average Spline and Arc Length Spline will always generate point UV's when you choose *Natural*.

If the primitive is open in both directions like a grid or a surface (so that the ends do not touch), then the advantage of vertex UV's does not apply since there are no matched seams on the single surface to worry about. But then again, if your RenderMan shader is expecting vertex UV's you can force the generation of vertex UV's.

Using vertex UV's gives you unique points at the closed seam whereas points are shared at seams and are, by default given a value of 0 for either U or V depending on the closed direction of the surface. If you want to make a closed surface open, simply insert a Carve OP in the chain and place a single carve in the surface of the direction that the surface is closed.

SCALE /su /sv /sw

Scales the texture coordinates a specific amount.

OFFSET /offsetu /offsetv /offsetw

Offsets the texture coordinates a specific amount.

ANGLE /angle

Rotates the texture coordinates the specified value.

Production Tip: Before applying a spline-based texture projection with the Texture OP, remap the U and/or V bases of the spline surface (using a Basis OP) between 0 and 1 to ensure a complete mapping of the texture. If a single texture map must be shared by several surfaces, the surface bases should be concatenated prior to being remapped.

FIX BOUNDARY SEAMS

Makes sure the texture wraps around correctly.
This is typically needed for wrapped polygonal topologies.

I49.3 INPUTS / GEOMETRY TYPES

INPUT I

Texture Source (accepts all geometry).

I49.4 USES / WORKS IN RELATION WITH

- Preparing a surface for a 2D image map (texture map) by assigning texture coordinates to the Source.

I49.5 SEE ALSO

- *Basis OP* p. 449
- *Rest Position OP* p. 726
- *UV Project* p. 838
- *UV Unwrap OP* p. 850

150 UV TRANSFORM OP

150.1 DESCRIPTION

The UV Transform operation transforms UV texture coordinates on the source geometry. Unlike the UV Edit operation, UV Transform supports animation of UVs.

UV Transform will automatically detect whether the input geometry uses point or vertex UV attributes, and will affect these attributes appropriately.

3D transforms are allowed in the UV Transform operation, as the UV attribute is actually 3 coordinates: U, V, and W. When used interactively, translates and scales are along the U/V axes while rotations are about the W axis.

150.2 PARAMETERS

GROUP

Subset of geometry to apply current operation to.

GROUP TYPE

The type of elements referenced in the Group field.

TRANSFORM ORDER

Order in which transformations occur.

ROTATE ORDER

Order in which rotations occur.

TRANSLATE */tx /ty /tz*

These three fields move the Source geometry in the three axes.

ROTATE */rx /ry /rz*

These three fields rotate the Source geometry in the three axes.

SCALE */sx /sy /sz*

These three fields scale the input geometry in the three axes.

PIVOT */px /py /pz*

The pivot point for the transformations (not the same as the pivot point in the pivot channels). The pivot point edit fields allows you to define the point about which geometry scales and rotates. Altering the pivot point produces different results depending on the transformation performed on the object.

PIVOT ABOUT GROUP CENTROID

Set the pivot to the centre of the input group.

SOFT RADIUS */rad*

Area of influence.

SOFT TYPE

Type of rolloff function.

TANGENT ANGLES */tandeg*

Angles of the cubic rolloff function's tangents. The first value applies to the tangent farthest from the source point, the second applies to the tangent closest to the source point.

KERNEL FUNCTION

Metaball kernel to use when rolloff Type is "Meta-ball"

METRIC

Measure of distance to use when performing a soft transform. XYZ corresponds to distance on the geometry. UVW corresponds to distance in UVW (texture) space. UV ignores the third dimension in texture space.

IGNORE POINT CONNECTIVITY

Affect only points connected to the points in the group, or all points within the radius.

IGNORE UV CONNECTIVITY

Affect only vertices connected to the vertices in the group as seen in the UV Viewport.

150.3 SEE ALSO

- *UV Edit OP* p. 833
- *UV Fuse OP* p. 836

151 UV UNWRAP OP

151.1 DESCRIPTION

UV Unwrap assigns texture UV coordinates to the Source geometry for use in texture and bump mapping.

Given a set of planes, this operation will project each input primitive onto the plane which will cause the least distortion in the shape. The planes can either be specified with one of the default sets or else provided through the second input.

In addition, polygons are grouped into regions which are connected polygons that map to the same plane. These regions are then laid out to prevent regions from overlapping.

151.2 PARAMETERS

GROUP

Subset of geometry to apply texture to.

PLANE GROUP

Subset of second input polygons to use as planes.

PLANES

Default plane orientations.

LAYOUT

How to lay the regions out.

SCALE

Type of scale to apply after the projection and layout.

SPACING */spacing*

Percentage of the laid out space to leave between adjacent regions.

ROTATE ORDER

Order of rotations.

ROTATE */rx /ry /rz*

Rotate angles in degrees.

151.3 SEE ALSO

- *UV Project* p. 838
- *UV Texture OP* p. 843

152 VERTEX OP

152.1 DESCRIPTION

The Vertex OP allows you to edit/create attributes on a per-vertex (rather than per-point) basis. It is similar to the Point OP in this respect. It supports two inputs, and will inherit the first input source by default.

There are currently three vertex attributes supported:
Diffuse Color, Alpha and Texture Coordinates.

When the attribute is defined, it can only occur on either points or vertices, but not both. Thus, if the input geometry has a point attribute for diffuse color, the attribute will automatically be “elevated” to be a vertex attribute (if diffuse colors are added in the Vertex OP).

The OP processes every vertex of every primitive. For each vertex processed, there are variables which allow you to know the:

- a) Vertex number of the primitive being processed
- b) The number of vertices in the primitive being processed
- c) The point which is referenced by the vertex
- d) The primitive which contains the vertex
- e) The total number of points
- f) The total number of primitives

There are also local variables to find out the values of some point attributes (i.e. position, normal – if they exist), in addition to vertex attributes.

152.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

KEEP / ADD / NO VERTEX COLOR

Select from the pop-up menu in order to pass through, create, or remove vertex color attributes from incoming geometry.

KEEP / ADD / NO ALPHA

Select from the pop-up menu in order to pass through, create, or remove vertex alpha (transparency) attributes from incoming geometry.

Using the local variables \$CA in an expression allows you to alter the existing vertex alpha attribute.

KEEP / ADD / NO VERTEX TEXTURE

Select from the pop-up menu in order to pass through, create, or remove vertex texture attributes from incoming geometry.

Using the local variables \$MAPU, \$MAPV, and \$MAPW in an expression allows you to alter the existing vertex texture attribute.

KEEP / ADD / NO CREASE

Select from the pop-up menu in order to pass through, create, or remove crease attributes from incoming geometry.

The “Crease Weight” attribute can be used to set individual edge crease weights for sub-division surfaces (see *SubDivide OP* - p. 777). This vertex attribute defines the weight for the edge which goes from that vertex to the next vertex in the polygon. For example, with a triangle (which has vertices 0, 1, 2), the attribute for vertex 1 defines the crease weight for the edge (1, 2). The attribute for vertex 2 defines the crease weight for edge (2, 0). The crease weight should be greater than 0. The larger the value for crease weights, the sharper the edge will be when sub-divided.

Crease attributes can be visualized by passing them into a Subdivide OP, or used for subdivision surface effects during rendering (*Object > Render page > Geometry > Polygons as Subdivison Surfaces* must be enabled in order for them to render).

I52.3 LOCAL VARIABLES

You can use all local variables of the Point OP – see *Local Variables* p. 671. Here is a subset:

N	The number of incoming primitives / points.
PT, NPT	Point number & total number of points.
PR, NPR	Primitive number & total number of primitives.
VTX, NVTX	Vertex number & total number of vertices.
CEX, CEY, CEZ	Centroid of the OP.
BBX, BBY, BBZ	Relative position of point within bounding box. Values are mapped between 0 and 1.
TX, TY, TZ	Point position.
NX, NY, NZ	Point normal directions.
MAPU, MAPV, MAPW	Point or vertex texture coordinates.
CR, CG, CB	Diffuse point or vertex colour.
CA	Point or vertex alpha value.

PT2 NPT2	Point number & total number of points for the second source.
CEX2, CEY2, CEZ2	Centroid of the OP for the second source.
TEX2, TY2, TZ2	Point position for the second source.

152.4 EXAMPLE

Example of manipulation:

```
vertex/cr = $BBX  
vertex/cg = $CG2  
vertex/cb = $CB - $CB2
```

152.5 SEE ALSO

- *Point OP* p. 667
- *SubDivide OP* - p. 777
- *Geometry Types > Attributes* p. 233

I53 VERTEX SPLIT OP

I53.1 DESCRIPTION

Takes an vertex attribute and splits any point whose vertices are off by more than that at that attribute.

For each point in the input geometry, Vertex Split tests the specified attribute to see if it is within tolerance on each vertex sharing that point. For each group of vertices outside the tolerance, a new point will be created. The vertex attribute can be then optionally promoted to a point attribute.

This operation is useful for promoting vertex attributes to point attributes without either creating a lot of points or destroying the seams in the vertex attributes.

I53.2 PARAMATERS

GROUP

The point group to act on.

ATTRIBUTE

The name of the vertex attribute to perform the test on.

TOLERANCE */tol*

The maximum error in any dimmension of the attribute before it will be split.

PROMOTE

Whether to promote the attribute to a point attribute.

I53.3 SEE ALSO

- *Primitive OP* p. 697

I54 VIEW OPERATION -

I54.1 MOUSE AND KEYS -

Space	Volatile Key
	Tumble the view.
	Dolly in-out (Perspective) or Zoom in-out (Ortho).
Alt 	Zoom camera in-out.
	Pan left-right and up-down.
Ctrl 	Rotate about the Z Axis.
Ctrl 	Box Zoom: Dragging left-right zooms in, dragging right-left zooms out.
Ctrl 	Pops-up the Operation menu.

I54.2 DESCRIPTION -

The View Operation allows you to directly manipulate your view of 3D geometry. You have controls to navigate the view through to anywhere in the scene using tumble, dolly, and pan controls.

I54.3 SUB-ICONS -

PARAMETERS BUTTON

Displays the parameters dialog for this Operation. Empty, because there are none.

I54.4 VIEWING OPERATION MENU (**Ctrl**)

HOME ALL **H**

Fits the geometry into the viewport, viewed head-on. Other elements may or may not be fitted into the Viewport.

HOME SELECTED **Shift H / G**

Fits the selected geometry into the Viewport, viewed head-on. Other elements may or may not be fitted into the Viewport.

HOME C-PLANE [A]

Fits the Construction Plane into the Viewport. The Construction Plane is displayed head-on with its normal pointing towards you. Geometry is not taken into account.

FRAME ALL [F]

Fits all geometry into the Viewport. Other elements, such as the Construction Plane, may or may not be fitted into the Viewport. The view remains unrotated.

FRAME SELECTED [Shift][F]

Fits the geometry of the current selection into the viewport. Other elements may or may not be fitted into the viewport. The view remains unrotated.

FRAME C-PLANE [Shift][A]

Fits the Construction Plane into the viewport. Other elements may or may not be fitted into the viewport. The view remains unrotated.

Homing vs. Framing: Homing brings things fully into view, but at the same time, it rotates them so that we look at them down a given axis. To bring things fully into view without rotating them, use one of the *Frame* commands.

TOGGLE ORTHO/PERSPECTIVE [O]

Switches the Viewport between Orthographic display and Perspective display.

TOGGLE SEE ONE / ALL OBJECTS [E]

Switches the Viewport between viewing only the current world Object being worked upon (avoiding the clutter of seeing too much) – or all world Objects in the scene simultaneously (so you can see the object in context).

SELECT VIEWPORT [X]

Selects between the four viewports when in Quad view mode.

PERSPECTIVE / TOP / FRONT / RIGHT / UV VIEWPORT [1][2][3][4][5]

Make the specified Pane the single dominant Viewing Pane.

TOGGLE SINGLE / QUAD [T]

Switches the Viewport between a singular or Quad-view display.

I55 VISIBILITY OP

I55.1 DESCRIPTION

The Visibility operation is used to hide and expose primitives in the 2D (UV) and 3D viewports.

I55.2 PARAMETERS

GROUP

The primitives to hide or expose.

ACTION */action*

Either "Hide Primitives" or "Expose Primitives".

APPLY TO */applyto*

If set to "Selected Primitives" the primitives in the group field are hidden or exposed. If set to "Non-Selected Primitives" all primitives except those in the group field are hidden or exposed.

VIEWPORT */viewport*

Either "3D Viewport" or "2D (UV) Viewport".

APPLY CUMULATIVELY */cumulative*

If this option is enabled, Visibility will append to the visibility information in the input geometry. For example, if primitive 2 was hidden in the input geometry and this primitive is to hide primitive 5, then both primitives 2 and 5 will be hidden in the output. If this option is disabled then only primitive 5 will be hidden in the output.

I55.3 SEE ALSO

- *LOD (Level of Detail) OP* p. 614

I56 WIREFRAME OP

I56.1 DESCRIPTION

This OP converts edges to tubes and points to spheres, creating the look of a wire frame structure in renderings. This is ideal for modelling tube structures and pipes.

I56.2 PARAMETERS

GROUP

If there are input groups, specifying a group name in this field will cause this OP to act only upon the group specified. Accepts patterns, as described in: *Scripting > Pattern Matching* p. 38.

WIRE RADIUS */radius*

Radius of the individual wires used in the construction of the geometry.

ROUND CORNERS

When selected, rounds the corners by placing spheres at the point locations with the same radius as the wires.

END CAPS

When selected, places end-caps on all wire geometry.

REMOVE POLYGONS

Removes the polygons from the output geometry, leaving only the converted line structures.

I56.3 INPUTS / GEOMETRY TYPES

All geometry types are accepted.

Note: Results will look best if you convert your geometry to polygons. Use a Convert OP before using this OP.

I56.4 SEE ALSO

- *Convert OP* p. 512

3 L-systems

I INTRODUCTION TO L-SYSTEMS

I.1 INTRODUCTION

Houdini has a number of powerful features which can produce very sophisticated geometry and effects. One of these is contained in the *L-system OP* p. 618. The L-system SOP provides a way to generate complex organic growth for simulating trees, lightning, and other branching structures.

BASED ON SELF-SIMILARITY

There are several factors which combine to organise plant structures and contribute to their beauty. These include: i) symmetry, ii) self-similarity, and iii) developmental algorithms. With L-systems, we are mostly concerned with the latter two. Self-similarity implies an underlying fractal structure which is provided through strings of L-systems. The property of self-similarity is described by Benoit Mandelbrot as follows:

“When each piece of a shape is geometrically similar to the whole, both the shape and the cascade that generate it are called self-similar.”

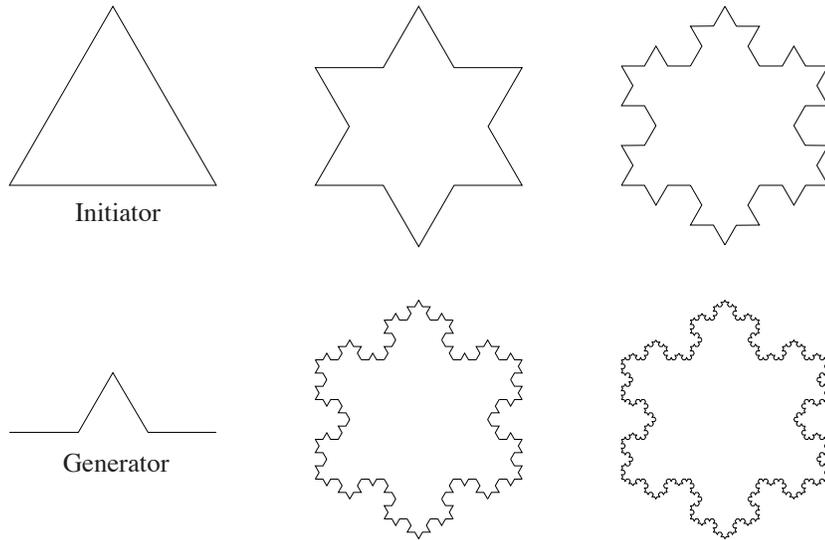
I.2 INITIATOR AND GENERATOR

L-systems provide a grammar for describing the growth of self-similar structures in time. L-system rules determine the underlying structures of growth in a way that is analogous to the way that DNA is thought to determine biological growth. This growth relies on the principle of self-similarity to provide extremely compact descriptions of complex surfaces.

The central concept of L-systems is that of rewriting. This works by successively replacing parts of an initial simple rule with a set of rewriting rules. Described by *Benoit Mandelbrot* as follows:

“One begins with two shapes, an initiator and a generator. The latter is an oriented broken line made up of N equal sides of length r. Thus each stage of the construction begins with a broken line and consists in replacing each straight interval with a copy

of the generator, reduced and displaced so as to have the same end points as those of the interval being replaced.



I.3 REWRITING GRAMMAR

In 1968, Astrid Lindenmayer introduced a string rewriting mechanism termed “L-systems”. The grammar of L-systems is unique in that the method of applying productions is applied in parallel and simultaneously replaces all letters in a given “word”.

The simplest example of a rewriting grammar is where two “words” or strings are used, built from the two letters: a and b , which may occur many times in a string. Each letter is associated with a rewriting rule. The rule:

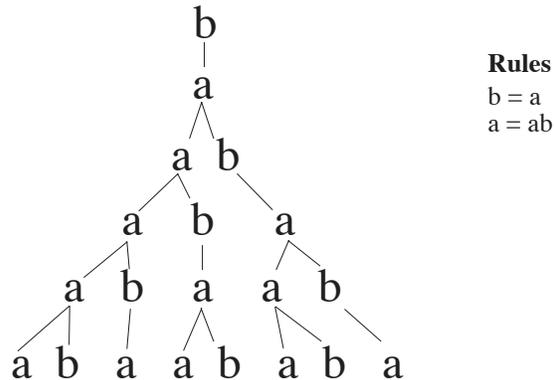
$$a = ab$$

means that the letter a is to be replaced by the string ab , and the rule:

$$b = a$$

means that the letter b is to be replaced by a .

If we start the process with the letter b, and follow it through in time, we see a certain pattern emerges by following the rewriting “rules”. Hence:



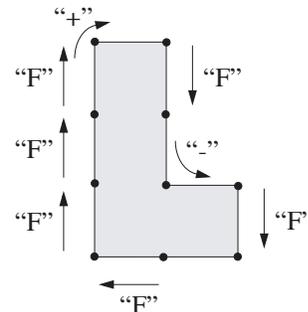
I.4 TURTLE INTERPRETATION OF STRINGS

The basic idea behind interpreting turtle strings is that a “turtle” exists at some point in cartesian space (XYZ), and in addition, the turtle also has a “heading” (angle). By giving the turtle a series of commands to follow in the form of character strings, we can tell the turtle to trace out various shapes. The simplest of these commands are:

- F Move forward a step, and draw a line connecting the turtles’ initial position and it’s new position.
- f Move forward a step without drawing a line.
- + (plus sign) Rotate the turtle by 90° so it faces right with respect to it’s current orientation. The angle can be changed, but we leave it at 90° to keep this illustration simple.
- (minus sign) Rotate the turtle by 90° so it faces left with respect to it’s current orientation.

With these simple rules, we can easily come up with a string that causes the turtle to draw a shape such as the letter “L”. For example, assuming the turtle is initially facing upwards, we would use the following string to create the letter “L”:

FFF+F+FF-F+F+FF

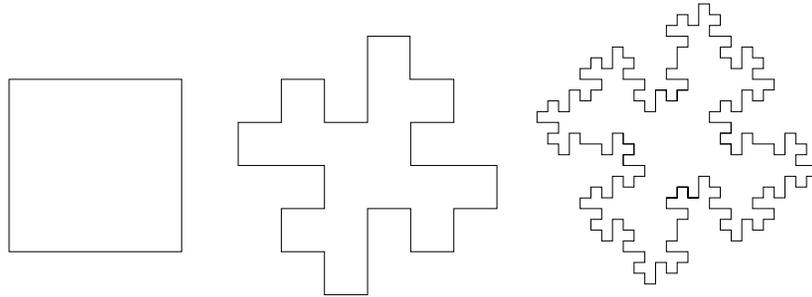


CREATION OF A KOCH ISLAND

This turtle interpretation of strings can be applied to strings contained in L-systems. For example, we can create an approximation of a quadratic Koch Island by interpreting the following strings iteratively (over and over again):

Premise: F-F-F-F
 Rule1 : F = F-F+F+FF-F-F+F

If you follow these through using the same principles of initiator and generator that we discussed several pages back, you should find, given the premise above, that the result of applying the *rule* repeatedly yields the following images:



By providing different rules, you can change the resulting shape. How to determine the shape is more complex, and we need to understand some things first before we can generate a shape that we predetermine.

Note: If you try keying this in, be sure your *Generations* is set to no more than 3.

1.5 CREATING OUR FIRST L-SYSTEM

Let us create our first L-system by defining the simplest of rules:

Premise: A
 Rule 1: A=F+A

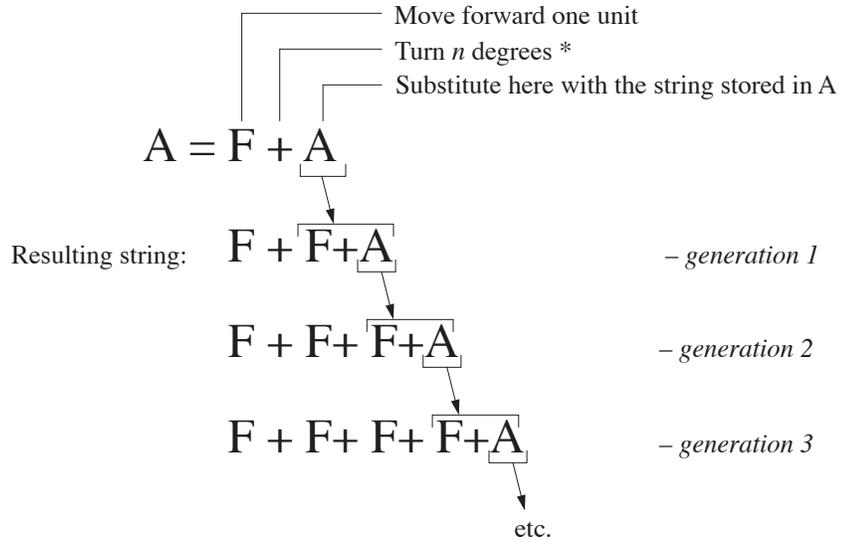
With this, we are saying “every time you see the letter ‘A’ replace it with the letters ‘F+A’”. This of course does not neatly resolve itself; it begs to be continued, because when you replace the letter ‘A’ with the letters ‘F+A’ you end up with another letter ‘A’ that needs replacing, and thereby, the process continues for as many iterations as desired.

So, the computer does this: It starts with the letter A (the premise). Then, it substitutes the letters ‘A+F’ into the A, because the definition for A contains an A in itself. The process is given a means to grow and extend itself infinitely (or for however many *Generations* are specified) in a process known as a *cascade effect*.

Note: Don’t confuse ‘F+A’ to mean F *plus* A in the regular mathematical sense. In this case, as turtle symbols, it simply means: move forward (the ‘F’), then turn left (the ‘+’), then proceed to continue movement of the turtle by moving through whatever instructions are stored in A (the ‘A’).

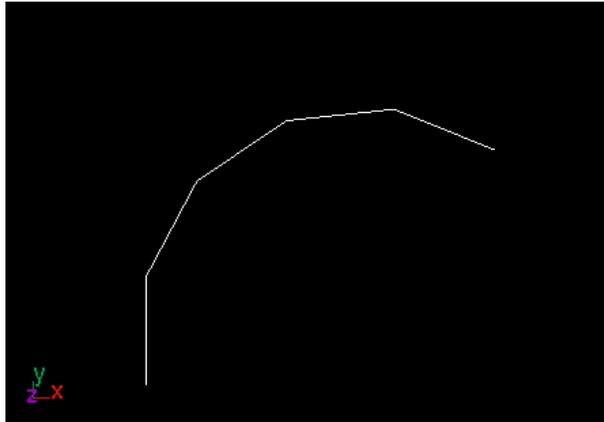
1.6 CASCADE EFFECT

Work the following out for yourself on paper:



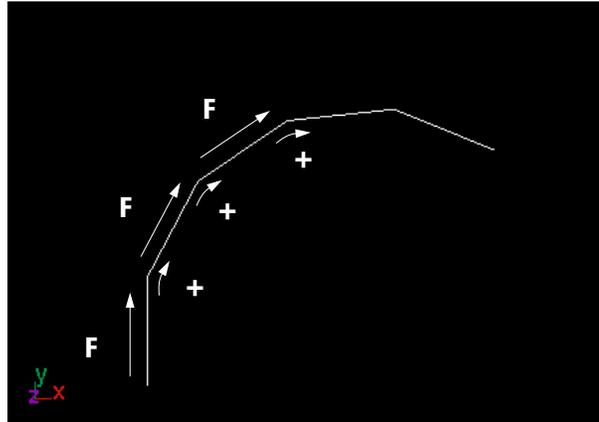
* The value of n can be set via the L-system SOP's Value/Angle parameter.

If we try this, we initially end up moving forward (F) and then turning several degrees (as defined in Values/Angle) to the left. This is then repeated several times (or generations). We end up with an endless series of $F+$'s. If we key this into the L-system SOP, and view the result in the viewport, we end up with the following:



Tip: Later, if we wanted to, we could randomize the angle produced by each “+”, and we’d end up with a squigly line whose straightness we can control by changing the amount of randomness in the angle. We could use this as the basis for creating a bolt of lightning (see *Lightning* p. 879).

Examine the resulting figure closely. What do you see? Why is it the shape it is? If we annotate the diagram, it may help us understand better:



As you can see, the result in the Viewport is a direct result of substituting ‘A+F’ everytime an ‘A’ is encountered in the L-systems rules for several generations. There are five generations shown here. Can you say why?

I.7 TRY THIS – TESTING THE L-SYSTEM

1. Place an L-system SOP in a Network Editor, and delete the default font1 SOP.
2. In the *Rules* page of the L-systems SOP’s parameters, delete all the default strings from the *Context Ignore*, *Premise*, and *Rule* fields.
3. In the *Premise* field, enter: *A*
4. In the *Rule 1* field, enter: $A=F+A$
5. Home the Viewport. Your result should be similar to the one in the illustration.
6. In the *Geometry* page, change the *Generations* parameter with the Value Grid to values between 1 and 10. Notice how the L-system grows and shrinks depending on the number of generations. When you are done, set the *Generations* to 7.
7. In the *Values* page, change the *Angle* parameter with the Value Grid to values between -45 and 45 (zoom out of the Viewport if necessary). Notice how the L-system grows in a different direction depending on the angle specified for the “+” turtle symbol. When you are done, set the *Angle* to 45.

Tip: You could use this behaviour as the basis for curling a sheet of paper, curling a scorpion’s tail, or any other number of animated effects.

2 BRANCHING

2.1 THE TURTLE SYMBOLS: [AND]

The L-systems so far are very nice, but they always end up being a single continuous line – not much like trees and plants which branch. In L-systems, the way we specify a branch is by using square brackets ([and]).

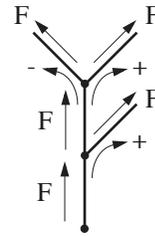
Whatever turtle symbols we place in brackets end up being executed separately from the main string under consideration. When the close bracket is encountered, the turtle returns to the place it was before the brackets were encountered.

Note: You may want to ensure that you have completely followed everything through up to this point by working everything out manually on paper. Things are about to get much more interesting, and it will become difficult to follow if you have only read the text without working it out for yourself.

2.2 SAMPLE OF BRANCHING

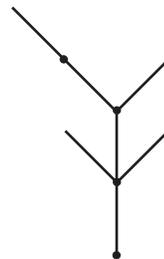
Instead of simply specifying a series of 'F' and '+' turtle symbols, we will try adding a few branches with the use of the '[' and ']' turtle symbols. Examine the figure below. Trace it out for yourself on paper using only the L-system string as a guide rather than copying from the illustration.

F [+F] F [+F] [-F]



See if you can come up with the L-system string that would describe this figure:

(enter your solution here)



You should have come up with a string like: F [+F] [-F] F [+F] -FF .

2.3 TRY THIS – BRANCHES

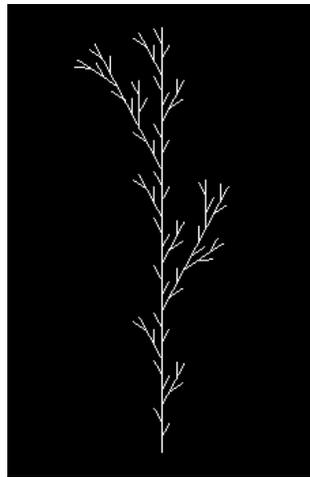
If we combine branching L-systems with iteration, we can come up with some very interesting geometry using only a few simple rules. Do the following:

1. Using the previously created L-system, change the default *Generations* to: 3.
2. In the same L-system, delete the existing rules, and enter these:

Premise: F

Rule 1: F= F [+F] F [-F] F

3. In the *Geometry* page, change the *Generations* parameter with the Value Grid to values between 1 and 5 (not more!). Notice how the L-system grows and shrinks depending on the number of generations. When you are done, set the *Generations* back to 3, and home the view.



4. In the *Values* page, change the *Angle* parameter with the Value Grid to values between 0 and 150. What do you notice when the *Angle* is set to 90?
5. When you are done, set the *Angle* to 22.5.

2.4 A DIFFERENT SET OF BRANCHES

Let's try changing the rules a bit, to see what else we can come up with. Here, notice how our branches contain three F's rather than just single F's – yielding a fuller bush-like effect.

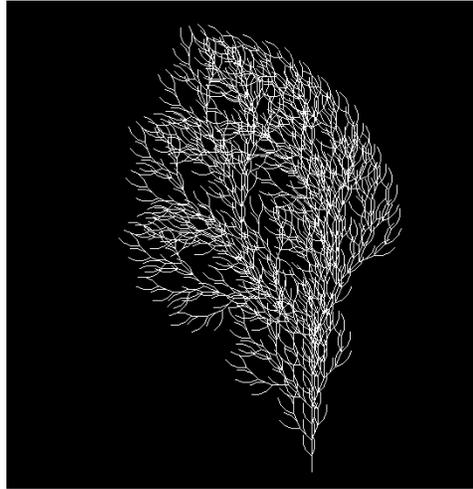
1. With the *Generations* set to 4, and the *Angle* set to 22.5°, enter these Rules:

Premise: F

Rule 1: F= FF- [-F+F+F]+ [+F-F-F]

2. Home the view if necessary.

Your result should resemble the image below:



This may look like a fantastic amount of geometry at first, but it works by replacing rule strings, just as we did in *Cascade Effect* p. 864. The difference is that a new extra L-system grows out the end of each branch (the 'F's get replaced by a whole new string of 'F's).

3. Set the *Generations* down to 3. Then, using the Value Grid in steps of 0.1, set the *Generations* to 2, and then to 1. See where all the geometry came from? It's just a repetition at each 'F' node of the basic shape in generation 1.
4. Try tumbling the view – we have a surprise – the bush is completely flat! In the two next exercises, we'll not only learn how to add volume to the plant, we'll also learn how to add leaves.

3 ADDING 3D VOLUME AND MULTI-PART RULES

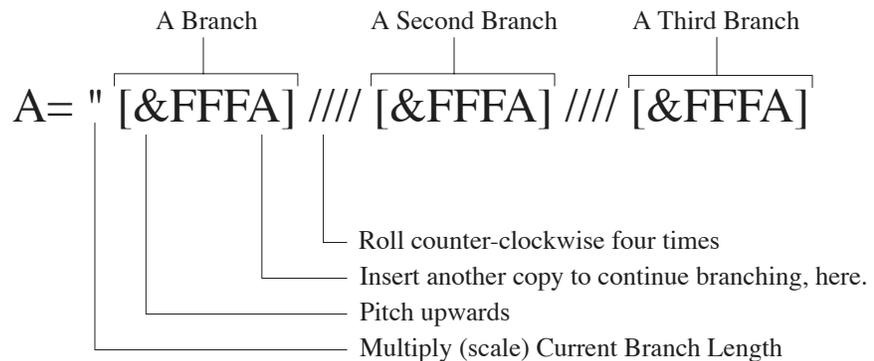
3.1 INTRODUCTION

Thus far, we have only created flat L-systems. In this exercise, we will learn how to add 3D volume to our L-systems, how to shorten branch lengths over time, and how to split up a long complex L-system string into multiple parts in order to make it more manageable.

3.2 EXERCISE – ADDING 3D VOLUME

Let's start with another L-system. This one is a bit more complex than the ones we have studied so far, but if we take it one step at a time, we should be able to understand what is happening. Do the following:

1. Place a new L-system SOP into a Network Editor, make it the display SOP, and delete all the default rules in the *Rules* page.
2. In the *Geometry* page, set the *Generations* to 1.
3. In the *Rules* page, enter the following rules:
 Premise: FFFA
 Rule 1: A= " [&FFFA] //// [&FFFA] //// [&FFFA]
4. View the result in the Viewport. You may have to home the view.
5. Set the *Generations* to 2. Tumble the view. Does this look somewhat familiar? Notice how a new three-branched section is added onto the ends of each branch made in the first generation.
6. Set the *Generations* to 3, and home the view. You should recognize the image from the default L-system. However, our rules are different than those of the default L-system. Here is a breakdown of our L-system:



Note: Notice how we use an 'A' at the end of each branch – this ensures that a new generation of growth based on the 'A' rule will begin at the end of each branch in the next generation.

ROLL ABOUT STEM – THE “ / ” SYMBOL

This breakdown gives us information on how to add 3D volume to an L-system. Instead of turning with the ‘+’ and ‘-’ symbols, we use the ‘/’ symbol to roll the turtle to change it’s heading about the stem created with the previous ‘F’ symbol.

You can see in this L-systems string, that between the three branches, we’ve insterted a number of Roll symbols (/) to space the branches about a common axis. To see the effect more clearly, do this:

7. Set the *Generations* to 2.
8. Remove the Roll symbols from Rule 1 – you should now see only one branch in the Viewport. Where are the other two branches? They are right on top of each other in the same place – without the instruction to roll about the stem, the three branches are no longer seperated from each other.
9. Insert just one Roll symbol between each branch so your string looks like this:
A= “ [&FFFA] / [&FFFA] / [&FFFA]
10. Tumble the view. See how the three branches start to seperate?
11. Add more Roll symbols between the branches until you are back to the original string (A= “ [&FFFA] //// [&FFFA] //// [&FFFA]).

THE PITCH SYMBOL – “ & ”

To stick with the sea-faring notation, the complement to a Roll is Pitch. This is provided with the ‘&’ symbol. To see the effect of pitch on the L-system, do this:

12. Remove the Pitch symbol from the first branch. Home and tumble the view.
13. Insert the Pitch symbol again, and observe how the branches spread out from the centre of the stem they branch from.
14. To get a feel for it, remove the Pitch symbol from each branch in turn, and then insert them back in again. Notice how the branches bunch towards the centre and then apart from the center when you put the pitch symbol back in.

Note: For a complete list of L-system turtle symbols, click on the small ? button in the L-system SOP’s Parameter Area.

3.3 EXERCISE – MULTI-PART RULES

As you can see, there is a fair bit of redundancy in our L-system string. For each branch, we repeat the string ‘&FFFA’. Wouldn’t it be nice if wherever we wanted to have a branch with ‘&FFFA’ in it, we could just type in another single letter.

We can. We can add another rule like this:

B= &FFFA

Then, wherever we want to use the string ‘&FFFA’, we can just type the letter ‘B’.

This allows us to simplify our rules somewhat. If we use the following rules:

Rule 1: A= "[B] /// [B] /// [B]"

Rule 2: B= &FFFA

then we can see that we have a short-cut for typing '&FFFA' simply by assigning that string to B, and using a 'B' wherever we want that string.

This has the added advantage of separating the rules for the branches from the main rule. We can change the branches independently of the main L-system string.

DO THIS – BREAKING IT UP

1. Change *Rule 1* to: A= "[B] /// [B] /// [B]", and add a new *Rule 2*: B= &FFFA
2. The L-system has shrunk. You may have to increase the number of generations to 5 or 7 in order to see it grown as far as previously. The reason for this is because it must work through an additional generation (to get to the repetition of the branches (the letter 'A')).

Because the branches are now separated from the main rule, we can easily modify all the branches at the same time. For example, if we wanted to decrease the length of the branches (less forward motion > less 'F' symbols in the branches) relative to the height of the trunk, we can simply remove a few 'F' symbols for the rule for the branches. Do this:

3. Change *Rule 2* to: B= &FFA. Notice how the entire model shrinks. This is due to the fact that the branches are now shorter relative to the trunk (the trunk uses three 'F's and the branches use only two).
4. Delete another 'F' from *Rule 2*, so it becomes: B= &FA. Observe how this changes the ratio of branch to trunk even further. When you are done, change *Rule 2* back to: B= &FFFA.

This gives us an idea how to change the length of branches relative to the trunk, or vice versa. We could increase the length of the trunk relative to the branches by adding a few extra 'F' symbols to *Rule 1*.

5. Change *Rule 1* to: A= "FFF [B] /// [B] /// [B]". Notice how the length of the trunk and branch stems are now longer relative to the 3-way splitting branches.

Note: The " (double quotes) symbol in front of the FFF will make them half their length. You can try deleting it to see the effect – the length of the stem doubles.

"FFFF = FF

3.4 EXPLANATION

From this exercise, we can see how to add 3D volume to a tree through the use of pitch and roll symbols within the L-system string. We also found out how to simplify repetitions within a string by creating another rule; and how to change the lengths of various components relative to each other by adding or deleting 'F' symbols within rules.

4 CREATING A SIMPLE PLANT WITH LEAVES

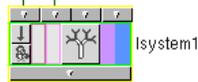
4.1 INTRODUCTION

In this exercise, we will take what we have learned about turtle interpretation of strings, branching, and iteration, and create a simple plant with leaves.

4.2 SETUP

SOP NETWORK

Place down an L-system OP, and set the parameters as noted below:



lssystem1
Geometry/Generations: 3
Values/Angle: 22.5

L-SYSTEM RULES

Delete the default rules from the L-system SOP, and enter these:

Premise: A
Rule 1: A= [&FA [fK]] ///// [&FA [fJ]] ////////// [&FA [fJ]]
Rule 2: F= S///// F
Rule 3: S= F

You will need to make the L-system SOP the display SOP.

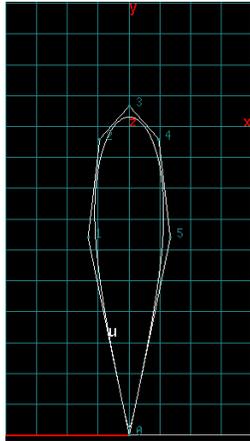
You will need to provide leaf geometry for the J and K inputs which are used in the rules – you won't see the result these have until you wire-in the leaf and flower geometry.

HOW TO INCREASE LEAF DENSITY

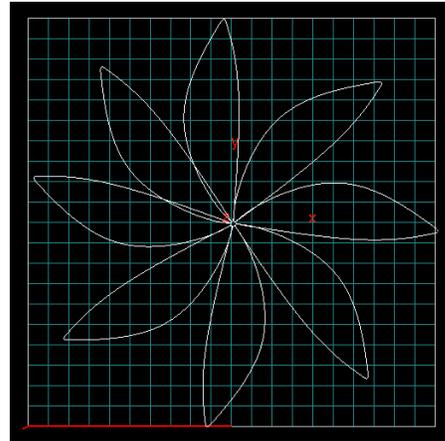
We won't do it for this example, but if you want leaves to occur along the branch length, then you need to make that part of a rule too. In other words, use something like: 'FJF' as opposed to just 'F'. This puts a leaf from the J input seemingly in the middle of a straight branch – however, this is not the way it actually works in real life. To get a more realistic look, decrease your branch length (length of an F), and increase the iterations – this gives you greater leaf density, but at the expense of speed.

MODELS FOR INPUTS I AND 2

Create the following geometry – the exact model isn't particularly important – so long as they look somewhat like a leaf and a flower. If they are the wrong size in proportion to the plant, we can always use a Transform SOP later to scale them appropriately.



model1

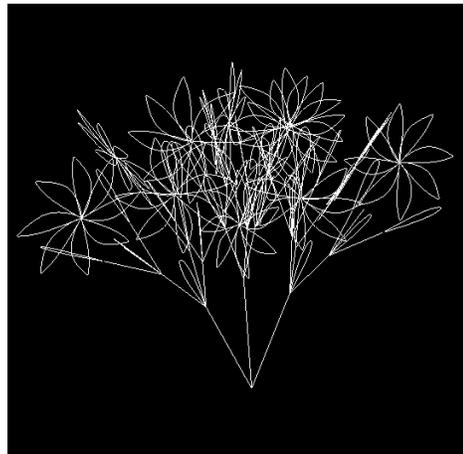


model2

Once you have modelled these, wire them into the first and second inputs of the L-system OP, and you should see petals and flowers appear within your L-system at those points we used J and K turtle symbols in our L-system strings.

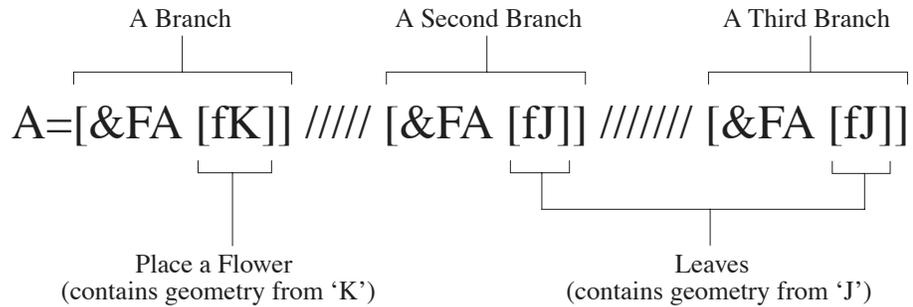
When we use the letters J, K, or M in an L-system string, it will place a copy of the geometry fed into the L-system's J, K and M Leaf inputs.

RESULT



4.3 EXPLANATION

Here is a breakdown the primary L-system string:



From this we can tell:

- This L-system has at least three primary branches
- The first branch will contain a Flower from the K-input
- The second and third branches will contain Leaves from the J-input

ABOUT THE RULES

This L-system is split up into three rules. The first rule is the *primary* rule. It is considered this by the fact that the premise is 'A', and the primary rule is the one that defines 'A'.

We can see this L-system has three main branches generated by the primary rule which creates three new branches from the apex of an old branch.

The branches of this small bush are quite similar to the one in the previous exercise. Each branch has the basic form: $[\&FA [fJ]]$. The & symbol provides for some pitch to the branch; then comes an edge *F* forming the initial node, an apex – *A* (which will subsequently create three new branches), and finally a leaf $[fJ]$.

The second and third rules specify internode growth. In subsequent generations, the internode gets longer and acquires new leaves. Between the branches, we have added a few / symbols in order to roll the three branches about the stem from which the current generation grows.

ADDING MODELLED GEOMETRY VIA THE LEAF INPUTS

The way we introduced modeled geometry into our L-system is through the use of the L-system SOP's J and K Leaf inputs. The way this works is that wherever we put the letter 'J' into an L-system string, it will place a copy of the geometry fed into the SOP's *Leaf J Input* (click on the input with to see the type of input).

Notice how a J or K leaf input is generally put at the end of a string (or branch). Anywhere you want a flower or a leaf in your L-system, simply insert a J or K.

We bracketed the J and K with $[f_]$ in order to advance the turtle forward a bit before we inserted the leaf geometry. A lowercase 'f' moves forward without creating geometry. If we didn't add an *f* together with the leaf, the leaf would be placed by it's centre, rather than it's stem.

5 BODHI TREE

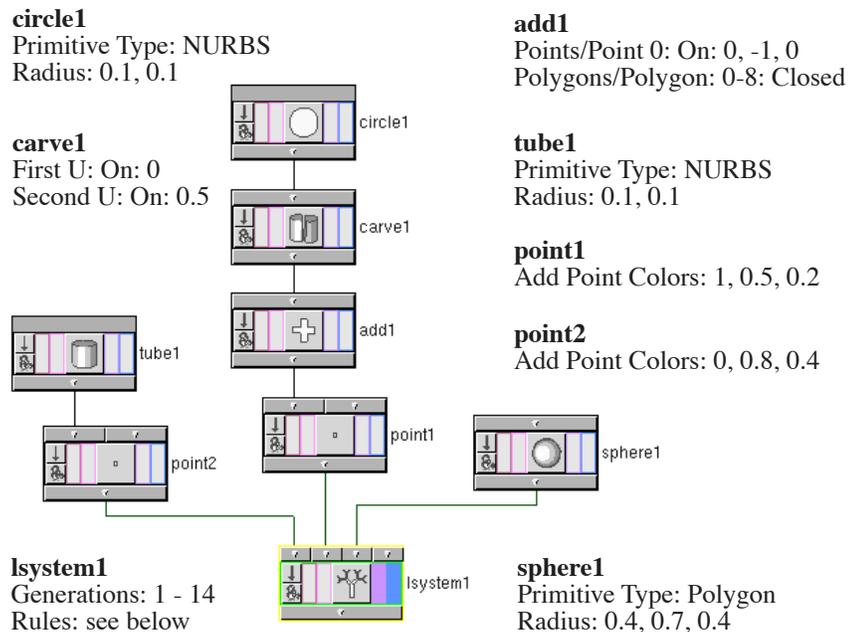
5.1 INTRODUCTION

It is often desirable to have different growth behaviours over the life of an L-system. In this example we will make a tree trunk grow with branches, then after several generations the trunk will stop growing and cactus-type leaves will begin growing, and finally after several more generations the leaves will bud with flowers and stop growing altogether.

5.2 SETUP

SOP NETWORK

Enter the SOP Editor, and create a SOP network as pictured below. The default parameters for each SOP are acceptable with the exception of those noted.



L-SYSTEM RULES

Delete the default rules from the L-system SOP, and enter these:

- Premise: A
- Rule 1: A: $t \leq 5 = fJB$
- Rule 2: A: $(t > 5 \ \& \ t \leq 10) = fK [+fKA] [-fKA]$
- Rule 3: A: $t > 10 = M$
- Rule 4: B = $[+fJ] [-fJ] A$

5.3 PROCEDURE

1. Make the L-system SOP the display SOP, and home the view.
2. Ensure the playbar is at Frame 1, and add a channel to the L-system's *Generation* parameter by selecting *Add Keyframe* from the  popup menu – then enter a *Generation* value of: 1.
3. Move the playbar to Frame 300, and enter a *Generation* value of: 14, and *Commit Change* from the  menu.
4. Click *Play*, and watch it grow (home the view if necessary). You should see:
 - The trunk grows from Generations 1.0 to 5.99
 - At Generation 6.0, the green branches start to grow until Generation 10.99
 - At Generation 11.0 - 11.99, the final Rule grows the White Flowers
 - After that, no more growth occurs.

5.4 EXPLANATION

The L-systems rules provide three different rules for *A*. There are also three different stages of growth in our L-system. Within each rule for *A* is a conditional statement which determines which rule for *A* applies in a given generation. What we effectively have are three sets of L-systems rules for the three different types of growth:

FOR TRUNK (GENERATIONS 1-5)

Rule 1: $A = fJB$

Rule 2: $B = [+fJ] [-fJ] A$

These rules are in effect when the Generations are less than or equal to 5 ($t \leq 5$). During this time, we move forward without drawing geometry (*f*), then place a twig from the *J* input (*J*), and then follow that with what is in the rule for *B*. In the rule for *B*, we have two branches (the $[]$ symbols). The first branch veers left ($+$), and the second branch veers right ($-$), after which it moves forward the turtle a bit (*f*), and then places another twig from the *J* input (*J*). Finally, it iterates the process by calling on the rule for *A* again (*A*). This process continues as long as the generations are less than or equal to 5.

FOR BRANCHES (GENERATIONS 6-10)

Rule 1: $A = fK [+fKA] [-fKA]$

This rule is in effect when the Generations are greater than 5 (i.e. 6), and less than or equal to 10 (i.e. will stop taking effect in Generation 11) ($t > 5 \ \& \ t \leq 10$). In this time, we see that we move forward (*f*), place a piece of geometry from the *K* input (*K*), then we have two branches, each of which will themselves continue branching (they have an *A* within them, so the rule will call itself). These branches veer to the left and to the right ($+ -$), and also use geometry from the *K* input (*K*). This process continues as long as the Generations (*t*) is greater than 5, and less than or equal to 10.

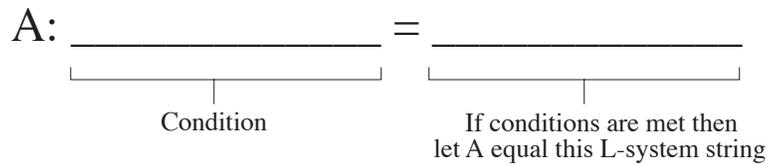
FOR FLOWERS (GENERATIONS 11-)

Rule 1: $A = M$

This rule is in effect when the Generations are greater than 10 (i.e. 11) ($t > 10$). Because the rule for A is simply equal to M - it places a copy of the geometry from the M input - and doesn't contain any way for it to grow (i.e. the symbol A is not within the rule for A), all growth beyond the eleventh generation stops. You will see the geometry from M "grown" for Generation 11.0 - 11.999.

COMBINED BY CONDITIONAL STATEMENTS

We need to slightly change this set of L-systems rules (growth behaviours) so they will be invoked based on the number of Generations of growth. Using a colon : within a rule means that it will evaluate that rule conditionally.

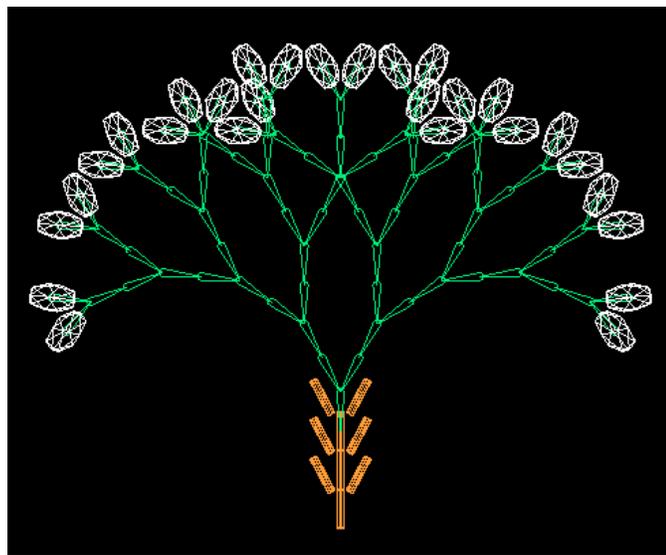


Conditional statements are regular mathematical expressions. To arrive at the final rules, we restate the rules so they will be evaluated based on the number of generations (t). We end up with the following rules:

- Rule 1: $A: t \leq 5 = fJB$
- Rule 2: $A: (t > 5 \ \& \ t \leq 10) = fK [+fKA] [-fKA]$
- Rule 3: $A: t > 10 = M$
- Rule 4: $B = [+fJ] [-fJ] A$

Tip: To make the flowers blossom independently of the tree growth, try using another L-system as an input for one of the leaf inputs.

5.5 RESULT



6 USING L-SYSTEMS TO GENERATE COPY TEMPLATES

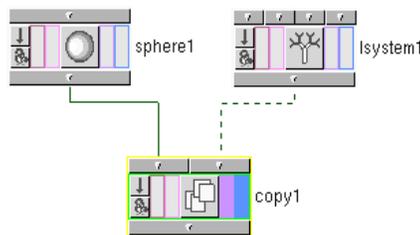
6.1 INTRODUCTION

A not so obvious use for L-systems is to create Copy SOP templates. This exercise demonstrates this feature.

6.2 EXERCISE

SOP NETWORK

Enter the SOP Editor, and create a SOP network as pictured below. The default parameters for each SOP are acceptable with the exception of those noted.



sphere1
Radius: 0.1, 0.1, 0.1

lsystem1
Geometry/Generations: 12
Values/Step Size: 0.3
/Angle: 0

L-SYSTEM RULES

Premise: A
Rule 1: A=F+A

Make the Copy SOP the display SOP, and you will see a small sphere copied to each node within the L-system – yielding spheres laid out in a line.

6.3 TRY THIS

1. Ensure the Copy SOP is the display SOP, and home the view.
2. In the parameters of the L-system SOP, in the Values page, change the *Angle* parameter between values of -30 and 30 using the Value Grid in increments of 1. You should see the string of spheres bend as the angle changes.

6.4 EXPLANATION

Instead of manually arranging the spheres, we have made our job easier by templating an L-system. This allows us parametric control of the bending. If you skinned the result and used a Copy SOP to place a copy at each point of a grid, you could create a sea of waving hairs, grass, or some other such creation.

7 LIGHTNING

7.1 INTRODUCTION

The fractal structure of lightning makes it conducive to simulation via L-systems. In this example, we see how to use an L-system to provide an electrical bolt of lightning to trace out the shape of a letter of the alphabet.

Using a Rails SOP allows us to scale and orient the generated L-system so it fits between two predefined points, giving us control over the start and end points of the lightning while still maintaining the inherent randomness of an electrical discharge.

7.2 SETUP

SOP NETWORK

Enter the SOP Editor, and create a SOP network as pictured below. The default parameters for each SOP are acceptable with the exception of those noted.

font1

Prim.Type: Bezier & Polygons
 Font: Times Roman
 Text: S

circle1

Prim.Type: NURBS
 Radius: 0.5, 0.5
 Centre Z: 1.0

extrude1 & extrude2

Fuse Points: Off
 Front Face: No Output
 Back Face: No Output
 Depth: 0, 0.009

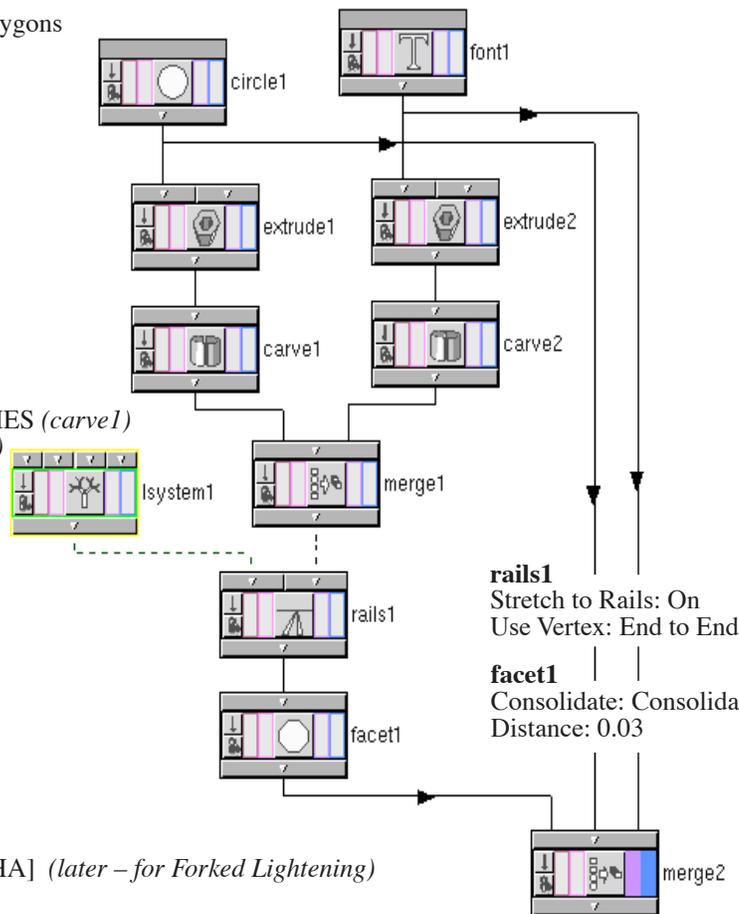
carve1 & carve2

First U: On: \$F/\$NFRAMES (*carve1*)
 First U: On: 0.25 (*carve2*)
 Operation: Extract

lsystem1

Geometry/Generations: 9
 /Random Seed: \$F
 /Continuous Angles: Off
 /Continuous Length: Off
 Values/Step Size: 0.22
 Rules1/Premise: FA
 /Rule1: A=~(30)FA

/Rule 2: F=FF[~(50)````HA] (*later – for Forked Lightning*)



7.3 EXPLANATION

The Bezier curve of the letter “S” is parameterized along the U direction. Using the expression $\$F/\$NFRAMES$ in the Carve SOP moves the point which the lightning strikes along the entire length of U (0...1) over the course of how many frames there are in the animation.

The lightning itself is generated by an L-system that works similarly to our original simple L-system: $A=A+F$ (see *Cascade Effect* p. 864). The difference is that here we pitch/roll/turn (using the $\sim(30)$ symbol) by a random amount up to 30° each generation instead of turning by a fixed amount (using the $+$ symbol). If you make it the display SOP, you will see it wildly convoluting from the single point from which the L-system originates.

Once we have our lightning, we need some control over where it goes. The easiest way to do this is to scale the end-points of the lightning to fit into the start and end points we want it to hit. In order to do this, we use the *Stretch to Rails* option in the Rails SOP which transforms and scales the Cross-section input (the lightning) so it lies between the two line segments generated by carving the font and the circle.

Making the Rails SOP the display SOP, you see the lightning bolt has been scaled and rotated so that it continuously connects the moving point generated by running from 0...1 in the Carve SOP and the point provided by the circle’s Carve SOP.

The continuous change in the shape of the lightning is due to the fact that we \sim (pitch/roll/turn) by a random amount between each “F” segment of the L-system. This gives us one set of random variations over the length of a lightning bolt. Try this:

1. Change the *Random Seed* to: 0. Click *Play*. Observe how the pitch/roll/turn between F segments is random, but the overall shape of the generated L-system is constant.
2. Change the *Random Seed* to: 1, and then to 2. This gives us another variation in the set of random pitch/roll/turns along the length of the generated L-system, but it’s still static.
3. Change the *Random Seed* to: $\$F$ (you may have to enable the *Expressions* button). The random seed is changed for each frame, providing continuous change both in the segment-to-segment pitch/roll/turns, but also from frame to frame.

The length of a single “F” is determined by the *Step Size* parameter. This is then scaled by the Rails SOP as necessary to make it fit between our two points.

We can add more irregularity to the lightning bolt by changing the *Geometry/Generations* parameter (try values between 3 and 20), and by changing the *Values/Step Size* parameter. Increasing the *Generations* produces more erratic lightning. You could also try animating the *Generations* from 3 to 20 to make it look like a power-surge trying to penetrate through shields.

We need to extrude the circle and font, because the Rails SOP can't handle one point polygons (what orientation would the rail be for a 1 point polygon?). It needs at least two points (a line) to map the cross-section onto. Because of this, rails will give us a ribbon of lightning that is as wide as the section we extrude from the Font and Circle SOPs.

To reduce the ribbon-strip back to a single polygonal line, we use a Facet SOP, which takes the ribbon of geometry generated by the Rails SOP, and whose thickness is determined by the Extrude *Depth*, and consolidates the points back into single thickness.

Finally, the Merge SOP takes the Font, the Circle, and the scaled and oriented L-system and merges them into a single output.

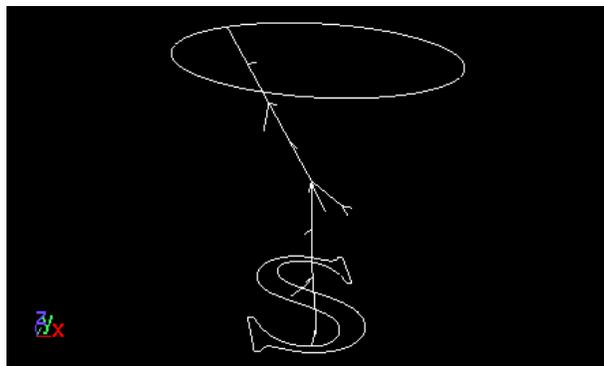
7.4 TRY THIS

FORKED LIGHTNING

If you want to add branching to the electrical bolt, we simply need to change the L-system that produces the lightning to include branching in it's rules. For example, you might try a rule something like:

```
Premise: FA
Rule 1: A=~(30)FA
Rule 2: F=FF[~(50)````HA]
```

What this does is redefines F to contain branches (the part in the square brackets is the branch off the main "trunk" of the lightning bolt). The Branch will pitch / roll / turn a random amount up to 50°, then it will create a short segment (one tenth of half an "F" – because of all the ' symbols), and then generate another iteration of branching at the end of that. The process for orienting and scaling the lightning with the Rails SOP is the same.



RENDERING

Use a Wireframe SOP appended to the Merge SOP. This provides the geometry to render. Apply a glowing material, and it is fairly simply to render the lightning. You may also want to use a Glow COP to add a glow around the lightning bolt after rendering.

8 HOW TO PRODUCE A DESIRED SHAPE – FERN

8.1 ASKING THE RIGHT QUESTION

The most common problem when people first start using L-systems is that they always want to work backwards to the natural method of L-system definition. They visualize a shape, and then want to know what the L-system string is to create that shape. This is exactly the opposite of how L-systems work. The only solution to finding the L-system that produces a preconceived shape is to truly remember the words of Mandelbrot, and understand the following:

“...each piece of a shape is geometrically similar to the whole.”

In order for you to find the L-system string to produce an overall desired shape, you need to build an L-system that iterates that shape self-similarly (fractaly) at each level of detail. This can be a difficult concept to understand at first, because the natural inclination is to think, “okay, so the shape generated by one iteration of the L-system should mimic the overall shape I ultimately desire...but how do I get it to look like *this*?” That is an entirely wrong way to go about thinking of these things. With L-systems you can’t be so rigid in defining the exact shape you want to end up with, you must learn to think in terms of recursive self-similarity. To end up with a shape generally like *this*, you must make a system that on a smaller iterative level mimics the overall shape you desire in a generalized way. When this overall form is iterated, the larger overall result will resemble (but may not precisely match) the initial generator.

8.2 SPECIAL CONSIDERATIONS

THINK OF SHAPES THAT ARE SELF-SIMILAR

To do this, one has to think of the different kinds of shapes that are self-similar. For example a building may not be self-similar, but a skyline is. A desk may not be self-similar, but rows of desks on several floors of a building are. A single flower budding may or may not be self-similar, but several leaves growing progressively larger along the length of a stem are. If the desired shape is unique and not recursively self-similar (e.g. a teapot or a telephone), then it may not be amenable to simulation via L-systems alone. Most of our creativity results from exploration of imposed order. Creativity in L-systems lies in exploring the possibilities inherent in the orderings of recursion.

GENERALIZE

Instead of trying to be more explicit about a shape, try to be less specific. Think of the overall gestalt of the form you wish to generate. Create an L-system’s rule to iterate this general form over and over again, thereby building up the general shape so it resembles the outcome of a single iteration of your L-system’s rule. For example, to create a fern leaf, we begin with an L-system string that roughly creates the shape of a fern leaf. You notice that the components, and sub-components of the fern

leaf mimic the overall shape of the leaf in detail. This is the way to achieve a desired shape with L-systems. Simple shapes work better than complex ones.

UNIQUE SHAPES MAY NEED TO BE PRUNED

If your desired shape is not one that is self-similar throughout itself, then straight L-systems may not be the best way to model your geometry. There is another way to produce an L-system of a desired shape through pruning. In general, this is a much simpler way of creating geometry in a specifically shaped envelope. Instead of trying to genetically alter the plant's shape to grow in a desired fashion, one simply allows the L-systems to fill a volume (defined with metaballs), and prune out the parts that grow outside of that volume – much like a topiarist trimming a hedge to conform to a desired shape. This technique is covered in *Pruned Growth* p. 891.

8.3 APPROXIMATING A FERN

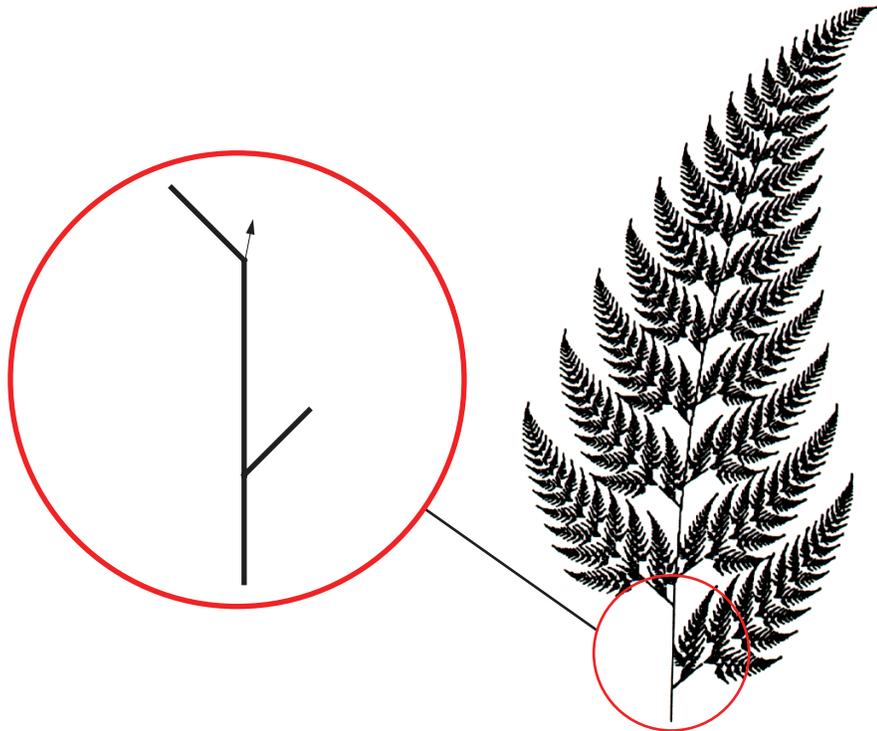
For this example, we are going to attempt to approximate the fern shape below.

After studying this shape, you will soon notice that each “leaf” of the fern is similar to the whole fern itself, and furthermore, each sub-leaf is also self-similar to the leaf. In order to produce a model that looks like this fern, we have to find the *essence of the form*.



A Fern leaf that we want to reproduce

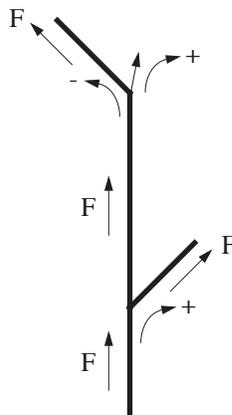
If we recall the idea of Initiator and Generator, and we study this form carefully, we can see that there is one essential shape that constitutes this underlying form we are trying to model. It is this:



This is the *essential shape* contained within the fern. The only way to find it for your own model is to study it carefully and determine the common shape that is repeated over and over within the form you are trying to replicate.

MAKE A GUESS OF THE RULES TO REPRODUCE THE ESSENTIAL SHAPE

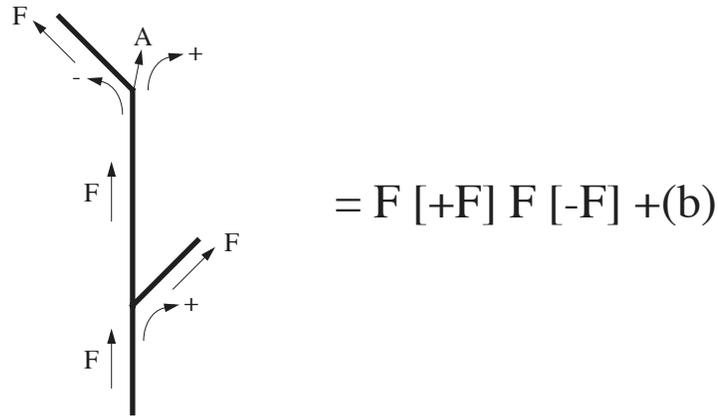
If our theory is correct, then we should be able to produce an L-system which will grow in the desired shape by iterating a set of turtle symbols that reproduces the essence of form.



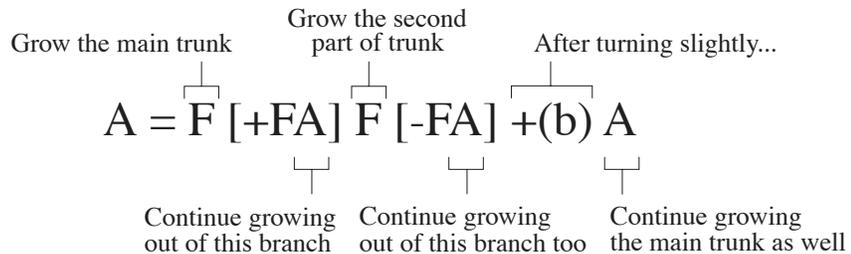
How to Produce a Desired Shape – Fern

We have derived from observation what we believe to be the essential form contained within the fern. Now we need to translate our sketch of this essential form into turtle symbols. This is relatively simple:

From the starting base, we need to move up a bit, so we use an ‘F’ turtle symbol. Then we need to branch once to the right, we can use the [] to do that part, then when we’ve returned from the branch, we need to move up again, so we use another ‘F’. Then we need another branch, so we use the same [] as we did for the first branch, only this time, it will turn in the negative direction (we’ll use ‘-’ instead of ‘+’). Finally, we need to turn the whole thing a few degrees before iterating the whole thing again (we’ll turn it by the number of degrees specified by the ‘b’ variable (look under *Values/Variable b*)).



Like *buds* on a plant, we want the growth to be iterated at the ends of the two branches and also along the main trunk. Adding the letter ‘A’ after the part that makes the branches (within the branch), and also after the part that draws the final part of the trunk is like putting a *bud* from which the whole thing is repeated over again (although slightly scaled down). So, our initial guess for us to work with should look something like:



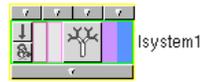
THE PREMISE

Once we have an idea of the rule we want to try, we will need to provide a “seed”. This seed is called the Premise. Our Premise here will simply be ‘A’. Once we have A, and we have the rule for developing A, it will work itself out from there. Let’s key this into the L-system SOP.

8.5 FINAL SETUP

SOP NETWORK

Enter the SOP Editor, and setup an L-system SOP as pictured below. The default parameters are acceptable with the exception of those noted.



Geometry/Generations: 5
 Values/Step Size: 0.25
 /Step Size Scale: 0.88
 /Angle: 45
 /Angle Scale: 0.618
 /Variable b: 4
 Rules/ see below

L-SYSTEM RULES

Premise: A

Rule 1: A= F [+H¹⁰A] F [-H¹⁰A] + (b) A

RESULT



5 iterations

7 iterations

9 iterations

Viewing our L-system in the Viewport confirms that we have extracted the correct essential shape from the image of the fern. If the generated L-system does not match our desired result, we would have to go back and study the original image more carefully to ensure that we find the correct essential form for us to iterate, and derive a new rule based on our observation.

9 DERIVING A PLANT

9.1 PROCEDURE

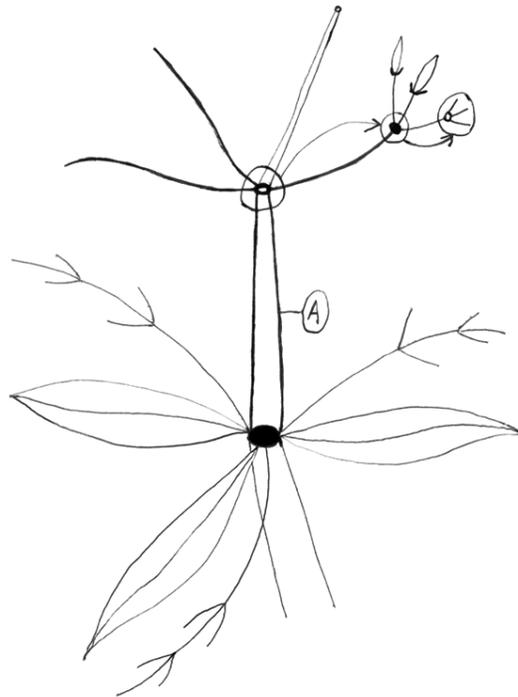
EXAMINE AND SKETCH THE PLANT

Wanting to create an L-system that will simulate the branching structure of an existing plant, we begin by a careful examination and sketch of the plant. We must pay careful attention to things such as:

- leaf phylotaxy (i.e. are there clusters of 2 leaves or 3 leaves per segment?)
- do the leaves alternate along the stem or trunk?
- every place where a branch starts, does it split, continue, and are there leaves?
- are there self-similar branching patterns like we found in the fern?

In the sketch below, you should be able to note at least several things:

- i) triple leaf coming out of the stem while it continues the ‘trunk’.
- ii) a self-similar branching takes place between the portion labeled ‘A’, and the branch-out to the right which has the three buds.
- iii) the three buds have the third one splitting up to continue further branchings.
- iv) there are two budding points – one continues along the trunk, and the other is in the third of the three-way splits along the trunk.



Sketch of the plant we are trying to reproduce.

MAKE A GUESS OF THE ESSENTIAL SHAPE

In the same way we derived the *essential shape* from the fern leaf, we will through observation try and find the simplest possible repeating form within the plant. We want to find the part of the plant that is repeated self-similarly, and the place where it buds-off to a repetition of this self-similar structure is noted with the symbol A .

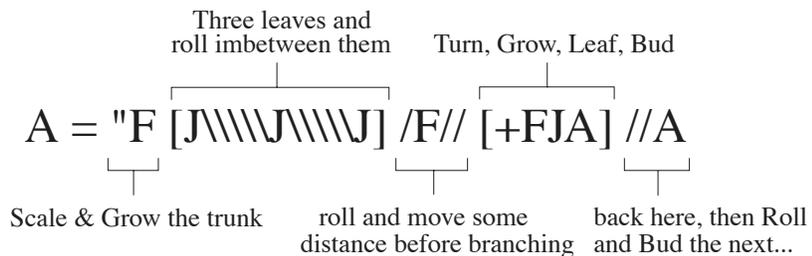


F [J/J/J] /F/ [+FJA] /A ???

After observing and studying the plant’s structure, we draw what we think is the essential shape contained within it as a schematic diagram (above). Once we have drawn this simplified schematic, if you have followed this tutorial all the way through (see *Sample of Branching* p. 866), then you can see that it is relatively easy to translate this schematic into its corresponding L-system turtle symbols. If our observation and guess are correct, then it will yield an L-system which will also look like the plant we are trying to reproduce.

DEDUCING THE L-SYSTEM STRING

After drawing up the schematic diagram, and getting the L-system string out of it, we key it into an L-system SOP, and see what happens. Of course, at first it doesn’t look right – so we add a few more roll (/) symbols for the leaves coming into the J input, and keep tweaking until the topology looks a bit more like the plant.

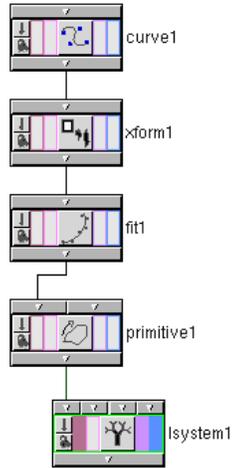


SETTING THE ANGLE

Even if we get the branching rules right, the system won’t look right if the branch angles don’t have the right values – and this may keep us from seeing if we’ve arrived at the right solution or not. Therefore, we periodically tweak the *Angle* parameter in the *Values* page, using the value grid to sweep through angles of 0 - 360. In this case, a value of 27.5 looks good – so we use it.

9.2 SOP NETWORK

Create the following SOP network, and adjust the indicated parameters:



curve1
 Coords: 0,0,0 -0.1,0.3,0 0.0,0.5,0.0 0.1,0.3,0,0
 Close: Enabled

xform1
 Rot-X: 30
 Scale: 1.6, 1.6, 1.6

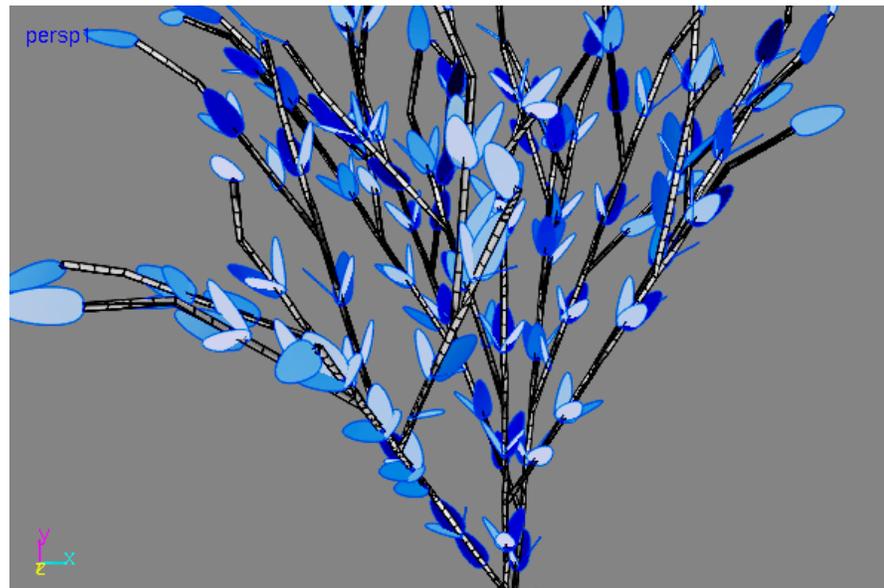
fit1
 Primitive Type: NURBS

primitive1
 Attributes/Add Colour: 0, 0.8, 0

lsystem1
 Generations: 6
 Values/Step Size Scale: 0.95
 /Angle: 27.5
 Rules/Premise: A
 /Rule 1: "F [J\\\\J\\\\J] /F// [+FJA] //A"

9.3 RESULT

That the output looks somewhat like our original plant confirms our derivation of L-system rules from what we observed of the branching structure in the actually growing plant. From here, its a matter of texturing and rendering the geometry.



10 PRUNED GROWTH

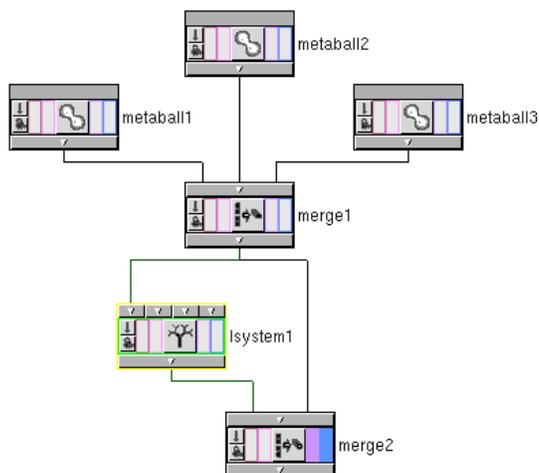
10.1 INTRODUCTION

One of the most commonly asked questions about L-systems is how to grow geometry within a bounded object. By using the Meta-test input of the L-system SOP, we can generate rules that will cause pruned growth – much like a topiartist carves an animal out of a hedge.

10.2 SETUP

SOP NETWORK

Enter the SOP Editor, and create a SOP network as pictured below. The default parameters for each SOP are acceptable with the exception of those noted.



metaball1
 Radius: 0.7, 0.7, 0.7
 Centre: -0.6, 1.0, 0
metaball2
 Radius: 1.0, 1.0, 1.0
 Centre: 0, 0.2, 0
metaball3
 Radius: 0.7, 0.7, 0.7
 Centre: 0.6, 1.0, 0
lsystem1
 Geometry/Generations:
 (\$F/\$NFRAMES)*13
 Values/Step Size: 0.045
 /Angle: 30
 Rules/See below

L-SYSTEM RULES

Enter the following rules for *lsystem1*:

Premise: FA

Rule1: A: in(x,y,z) = F [+FA] -FA : 80

Rule2: A: ! in(x,y,z) = A%

Important! If the L-system is not initially within the metaball envelope, it will be dormant – it will be unable to grow. If you reposition any of the Metaballs, be sure to add a Transform SOP to move it's origin to be within the metaball envelope.

FURTHER INSTRUCTIONS

When you have entered all the parameters do this:

- Set the Playbar to frame 1, and click *Play* to watch the volume grow.

Note: Do not set Generations parameter beyond 13, or the volume of geometry will become incredibly dense.

10.3 EXPLANATION

Although this L-system is very much like any standard L-system, it has the unique attribute of checking to see if the next iteration of growth will be within the Meta-test bounds and, if not, it prunes the current branch.

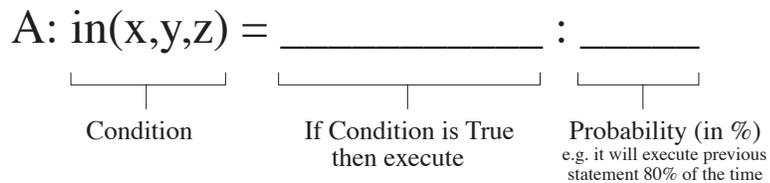
The merged metaballs in this example provide the shape or “envelope” for the pruned L-system to grow in.

Once a bounding volume of metaballs is defined, we feed that into the L-system Meta-test Input, and the L-system rules take care of testing to see if the growth is within those bounds.

HOW DOES THE L-SYSTEM CHECK IN-BOUNDS?

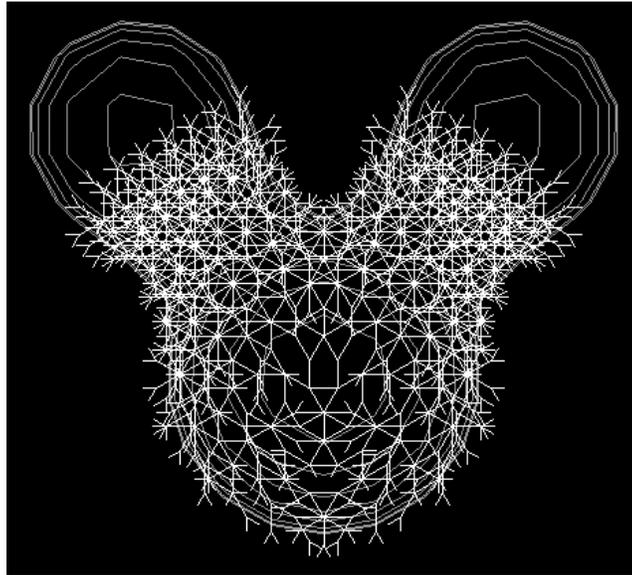
The way the L-system rules test this is via the conditional statement in Rule 1. The colon (:) after the *A* means that it will conditionally evaluate the rule for *A*. The condition is: *in(x,y,z)*, which asks if the branch *A* is growing within the Meta-test input bounds. If so, then it extends the branch another iteration with the rule that follows the = sign. This is followed by a probability of rule execution if the condition is true. In this case, the probability of the rule being executed is 80%. We can use an expression here instead of a number (80) to make the probability change as the number of generations increases.

If the branch *A* is not within the bounds, then it proceeds to the next rule which is also conditional. This time the condition is *! in(x,y,z)*. This means the rule will apply if the growth is *not* (the ! symbol) within the Meta-test input bounds (*in(x,y,z)*). If so, then the growth for that branch at that iteration should be pruned – the *A*% prunes the growth.



* Any symbols in the Condition and Probability statements are mathematical, while anything after the equals (=) are turtle symbols.

RESULT



HOW WE GOT 13 GENERATIONS OUT OF 300 FRAMES

For the *Generations* parameter, we used the expression $(\$F/\$NFRAMES)*13$. What this does is takes the (CurrentFrame ÷ TotalFrames) which normalizes the frame number into the range of 0.0 - 1.0, and then multiplies that by 13 – the total we want to reach when we reach the last frame (frame 300).

10.4 TRY THIS

BE YOUR OWN TOPIARIST

Create a new shape with metaballs, and use this as the growth-envelope. Place the L-system within the new shape, and watch the geometry grow within it.

It is absolutely necessary that the L-system ‘seed’ be within the metaball envelope, or it will remain dormant and unable to grow.

Pruned Growth