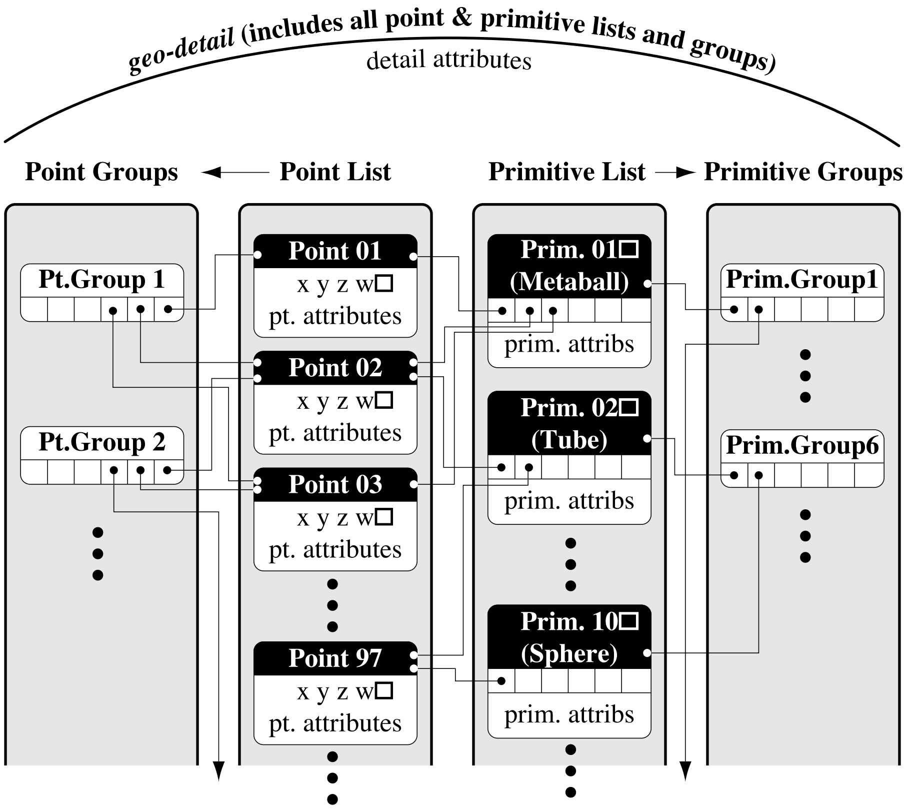
I Geometry Types

1 GEOMETRY DETAIL

The basis for all geometry creation lies in the knowledge of the different geometry types that Houdini is capable of producing.

1.1 THE GEO-DETAIL

Geometry in Houdini is stored in the data structure called a *geo-detail*. A geo-detail is a comprehensive listing of the entire geometry model that exists within Houdini at a given time. When you save your work into a *.geo* or *.bgeo* file, it records the entire geo-detail.



The geo-detail contains a point list, a primitive list, point groups, and primitive groups. They are discussed in detail in the following section. The detail manages attributes per point, vertex, and primitive, and its own *detail attributes*.

For a detailed breakdown of the .geo file format, see *.geo File Format Description* p. 281 in the *Formats* section.

2 THE POINT LIST

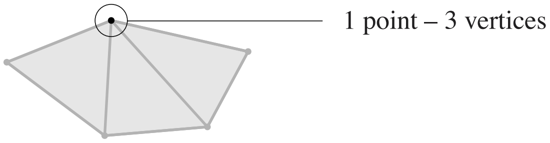
The geo-detail stores all points – both those attached to primitives and those that are free-floating – in a list. These points are simply X Y Z W locations in space. The W component is the spline weight (not mass, but the amount of pull on a spline hull). Each point can also contain attributes like colour, normal, and mass (see *Attributes* p. 233).

Every point in the point list can be referenced by one or more primitives in the primitive list. For example, a sphere primitive may reference a point in the point list as its definition for the location of its centre, and the same point might also be the control vertex of a NURBS curve.

WHAT IS THE DIFFERENCE BETWEEN POINTS AND VERTICES?

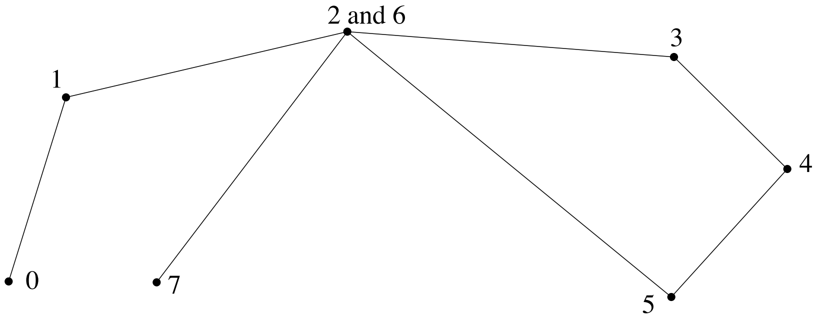
The difference between points and vertices is that a point can be shared between primitives while vertices are unique. A point is simply "a place in space" as defined by four numbers (X, Y, Z, W).

A vertex on the other hand is a reference to a point. Primitives use vertices to reference a point (e.g. the nodes of a polygon, the centre of a sphere, or the control vertex of a spline).



For example, if three polygons have one of their vertices in exactly the same place, *and* share the same point in the list, that place will contain three vertices, even though it is only a single point in the point list. Similarly, each vertex may reference a unique point, even though the points coincide in space.

It is also possible for certain primitives to use a point more than once.



A point being reused: seven points and eight CVs

To sum up, the vertices of a primitive are always unique, while the points they reference might be shared between one or more primitives in the geo-detail.

3 THE PRIMITIVE LIST

The primitive list may contain any number of primitives of the following types:

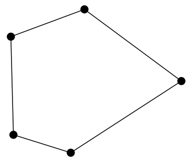
3.1 POLYGONAL

POLYGONS

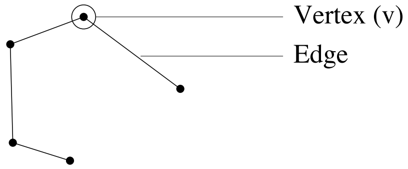
Polygons are shapes constructed from a series of straight edges. These edges are defined by a series of vertices.

types of polygons

a) Closed or Open



Closed Polygon



Open Polygon

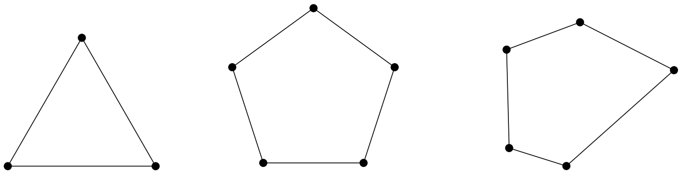
A closed polygon shares its first and last vertex and is flagged internally as "closed". Thus, if an open polygon has five vertices, it will still have five vertices when closed. The last (closing) vertex is only implied.

b) Planar or Non-planar

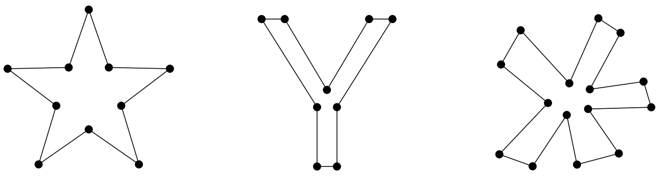
Planar polygons are those whose vertices lie in the same plane in 3D space.
Non-planar polygons have vertices that do no lie in the same plane in 3D space.

c) Convex or Concave

A polygon can be convex or concave, as illustrated below:



Convex Polygons

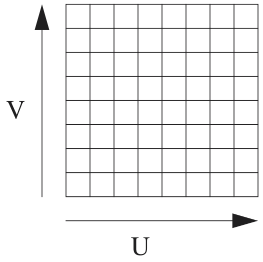


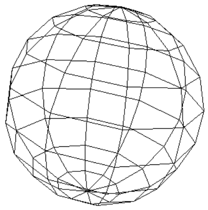
Concave Polygons

A polygon is convex if any vertical or horizontal axis intersects it at most twice.

3.2 MESHES

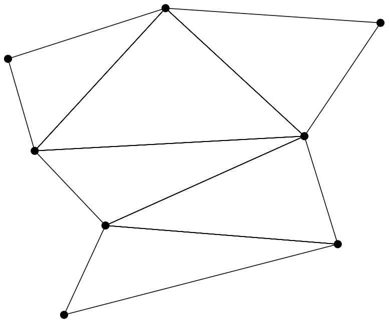
Meshes are a collection of edges and vertices that can be represented as having a number of rows and columns based on a UV co-ordinate system. They can be modified into various shapes such as tubes and spheres by changing the point coordinates and/or closing the mesh in U or V, while maintaining their row/column-like topology. For example, below right is a mesh modified into a sphere by wrapping the mesh in U.



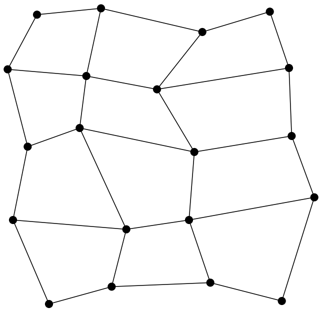


Both primitives have the same $m \times n$ point topology, only the point coordinates are different. What looks like individual polygons in the above figure are actually intrinsic parts of the primitive.

A figure that doesn't have an $m \times n$ topology cannot be a primitive mesh.



This is not a primitive mesh, because
it does not have an *m x n* topology



This is a 5 x 4 primitive mesh.

3.3 SPLINES

Houdini allows you to create both Bézier and NURBS curves and surfaces. Refer to the section *Splines* p. 238 for a complete discussion of these types.

3.4 QUADRICS

ELLIPSE (CIRCLE)

These are circles whose X and Y radii are specified independently of each other. Ellipses are stored as primitives rather than as polygons, NURBS, or Bézier curves. They contain their own set of parameters: *centre xyz*, *x-radius*, *y-radius*, and a 3×3 rotation matrix which determines which direction the primitive faces. If both radii are equal, the ellipse is known as a circle.

Ellipses have only one vertex – their centre. They are very light (in terms of data) objects, so avoid building them as polygons and splines unless it is necessary.

ELLIPSOID (SPHERE)

Ellipsoids are the three-dimensional analogue of an ellipse. They are defined by the parameters *centre xyz*, *x-radius*, *y-radius*, *z-radius*, and a 3×3 rotation matrix which determines the orientation of the ellipsoid. The ellipsoid is known as a sphere if all three radii are equal.

Ellipsoids have only one vertex – their centre. They are very light objects, so avoid building them as polygons and splines unless it is necessary.

TUBE (CYLINDER)

Tubes are primitive types which resemble cylinders, with the exception that the upper and lower diameter can be changed independently of each other. They also have the ability to have “Caps” – coverings over their end surfaces.

Tubes are defined by a *centre xyz*, a *top radius*, a *bottom radius*, a *height*, and a 3×3 rotation matrix which determines the orientation of the tube. Tubes degenerate into cones if one of the radii is zero.

Tubes have only one vertex – their centre. They are very light objects, so avoid building them as polygons and splines unless it is necessary.

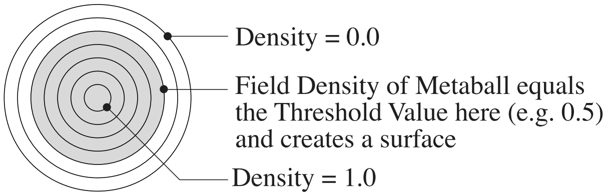
3.5 METABALLS

Metaballs can be thought of as force fields whose surface is an implicit function defined at any point where the density of the force field equals a certain threshold. This field can currently be specified as an elliptical or super-quadric shape around a point. When two metaballs overlap in space, their field effects are added together.

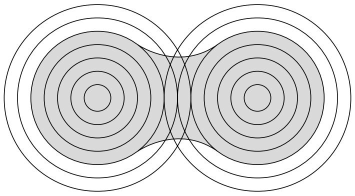
The field is specified by a weight and a kernel function. The kernel function results in a value of 0 at the outside edge of the metaball and a value of 1 at the centre. The kernel function is scaled by the weight to shift the location of the surface closer or further away from the centre.

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:


Density = 0.0
Field Density of Metaball equals the Threshold Value here (e.g. 0.5) and creates a surface
Density = 1.0

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.



Metaballs are defined by the parameters *Centre x/y/z*, *Radius x/y/z* , *Exponent x/y/z* , and a 3 × 3 rotation matrix which determines the orientation. A metaball is known as a super-quadratic if either exponent is not equal to one.

You can see a metaball's sphere of influence by turning on Display Hulls in the Viewport Options dialog (see *Primitives Display* p. 124 in the *Interface* section).

In the Model Editor, a metaball can be selected only by its hull.

PUSHER METABALLS

It is possible for metaballs to have negative Weights ("Pusher" Metaballs). This allows holes to be created by effectively subtracting from the surface.

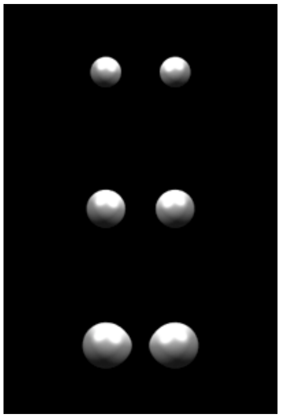
WHAT DOES AN EXPONENT DO?

In the instance of metaballs, the XY and Z exponent determines the inflation towards "squarishness" or contraction towards "starishness" as described below:

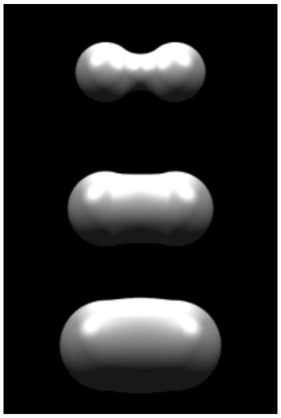
Value > 1	Results in metaballs that appear more like a "star".
Value < 1	Results in metaballs that appear more "squarish".
Value = 1	Results in metaballs that appear spherical.

In Houdini, metaballs are often used as force fields for particle systems (see *How Force Fields Work* p. 251). You can create metaballs with a Metaball SOP, or in the Model Editor.

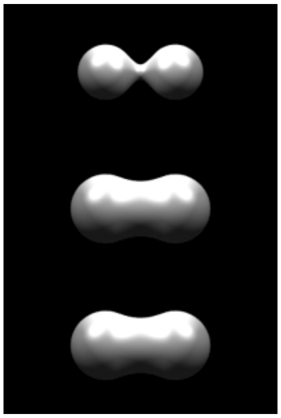
METABALL MODEL TYPES



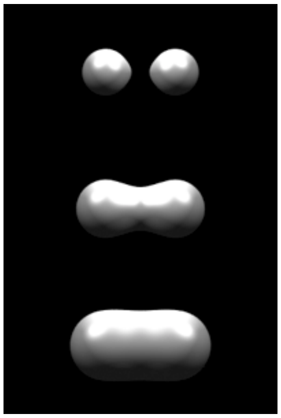
Blinn



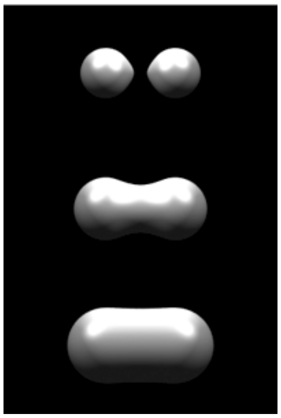
Elendt



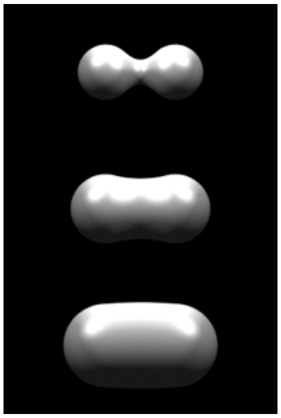
Hart



Links



RenderMan



Wyvill

blinn kernal

Always puts a sphere at the blob centre, even if the weight is less than 1.0.
The Blinn model is the fastest and most stable of all the models.

wyvill and elendt kernals

These models are very similar; only the weight distribution function is different.

links kernal

This is the slowest method, but provides a good compromise between the Blinn and Wyvill methods in terms of weight distribution.

renderman kernal

This kernel is used by Pixar's Renderman (and possibly other RIB renderers).
It is defined as: *Density = 1 - 3*R^2 + 3*R^4 - R^6* .

hart kernal

This is a kernal function suggested by the mathematician, John Hart.

3.6 PARTICLE SYSTEMS

A particle system consists of a group of discrete particles which change over time. Each particle has its own attributes controlling size, position, velocity, etc. They can generate new attributes depending on their age, or they may die. Assigning values discretely to each particle enables realistic modelling of systems involving turbulence such as: smoke, wind, fire, dust, and hair.

You can use any point or set of points as the basis for the particles in a particle system. Grid or Sphere SOPs are often employed for this purpose. You can set the particle system into motion by applying POPs or a Particle SOP. You can then subsequently influence the particle system with a Force SOP.

See the Particles (POPs) section of the manual for more information.

4 POINT GROUPS

Point groups are collections of associated points that are treated as a set. This is important because you can apply operations to them as a whole – saving you from having to apply discrete operations to each element. Furthermore, point groups can be used to filter-out points not needed for a specific operation; rather than affecting an entire input, grouping restricts the scope of an operation to the points in a group.

4.1 CREATING POINT GROUPS

To group points in Houdini use the Group SOP (see *Group OP* p. 592 in the *Editing Geometry* section), any SOP with a point group input field, or the Select state in a Viewport.

To group points in the Select state:

1. Use the Select state, and in the sub-icons, choose the Point Groups icon.
2. Select the desired points with the cursor.
3. Call up the Parameters dialog by clicking the ✙ button, and click on the *Combine Groups* page-tab.
4. Type a new group name in the edit field, and type Enter.
5. Click on the *Group <– Selection* button.

4.2 ORDERED AND UNORDERED GROUPS

A point group can be ordered or unordered. In the Model Editor's Select state, a single click of the mouse button performs an ordered selection. Bulk selections are made by dragging the cursor across the points. This action creates a marquee box that encloses a number of points. Points selected in this fashion generate an unordered group.

The only time bulk selections generate or maintain an ordered selection is when only one point is caught in the marquee box. Unordered groups store their points in creation order; ordered groups store points in selection order.

If you want to reselect the points in the group, you can do so by calling up the Parameters dialog from the Select state, and selecting the group name from the ▷ pop-up menu under the *Combine Groups* page-tab. Then click on the button *Selection <– Group*.

When a point is deleted, Houdini automatically removes the point from all the point groups it might belong to.

5 PRIMITIVE GROUPS

Primitive groups are a collection of associated primitives that are treated as a set. This is important because you can apply operations to them as a whole – saving you from having to apply discrete operations to each element. Moreover, primitive groups can be used to filter-out primitives that aren't needed for a specific operation. Rather than affecting an entire input, grouping restricts the scope of an operation to the primitives in the group.

5.1 CREATING PRIMITIVE GROUPS

To group primitives in Houdini use a Group SOP (see *Group OP* p. 592 in the *Editing Geometry* section), any SOP with a point group input field, or the Select state in a viewport.

To group primitives in the Select state:

1. Use the Select state, and in the sub-icons, choose the Primitive Groups icon.

2. Select the desired primitives with the cursor.

3. Call up the Parameters dialog by clicking the ✙ button, and click on the *Combine Groups* page-tab.

4. Type a new group name in the edit field, and type Enter.

5. Click on the *Group <– Selection* button.

5.2 ORDERED AND UNORDERED GROUPS

A primitive group can be ordered or unordered. In the Model Editor's select state, a single click of the mouse button performs an ordered selection. Bulk selections are made by dragging the cursor across the primitives. This action creates a marquee box that encloses a number of primitives. Primitives selected in this fashion generate an unordered group.

The only time bulk selections generate or maintain an ordered selection is when only one primitive is caught in the marquee box. Unordered groups store their primitives in creation order; ordered groups store primitive in selection order.

If you want to reselect the primitives in the group, you can do so by calling up the Parameters dialog from the Select state, and selecting the group name from the ▷ pop-up menu under the *Combine Groups* page-tab. Then click on the button *Selection <– Group*.

When a primitive is deleted, Houdini automatically removes the primitive from all the primitive groups it might belong to.

6 ATTRIBUTES

6.1 INTRODUCTION

Attributes include information about an entity such as its color, velocity, normal, and so on. There are many different attributes. Some of the most common ones are discussed below. Attributes can be attached to vertices, points, primitives, or the whole geometry. Since there are usually more vertices than points, having a vertex attribute will consume more memory than a point attribute. Similarly, since there are usually more points than primitives, having a point attribute will consume more memory than a primitive attribute. However, point attributes are interpolated across primitives, allowing more local flexibility than primitive attributes (e.g. color). Also, vertex attributes deal with the situation where shared points need different values for the attributes, like the seam of a polar texture map for example.

6.2 TYPES OF ATTRIBUTES

There are three different attribute data types. Each is handled slightly differently internally.

Vector Data	This data type represents a 3D vector in space. When any transforms occur on the detail, this attribute will also be transformed. Examples of a vector attribute are normals (N) or velocity (V).
Floating Point Data	This data type represents an array of floating point values. The values are not transformed when the geometry gets transformed. Some examples of this type of attribute are diffuse colors (Cd), and texture co-ordinates (UV).
Indexed String Data	This attribute consists of an ordered list of character strings. The attribute stored with the element is an integer representing the offset into the array of strings. A value outside the bounds of the array is considered to be “not assigned”. An example of this is the material attribute.

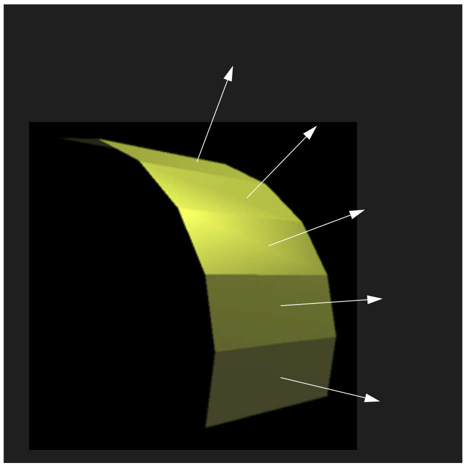
The following are examples of the most commonly used attributes.

6.3 NORMALS

A normal is a directional vector associated with a particular geometric entity, commonly perpendicular to it. The normal to a surface at a given point is a vector perpendicular to the surface at that point, and is computed as the cross product of the tangent vectors at that point.

The direction the normals take (up or down) is dependent on the order in which the cross product is computed (imagine a cork moving up or down depending on the direction the cork screw turns).

Normals are used for such things as: the basis for the direction things move over time, and for determining shading. In the Model Editor you can use point and primitive normals to pick, and even translate geometry along the normal.



Surface normals indicate the direction a surface faces. This is used to determine the amount of shading that a surface receives; the more it faces the light, the lighter the shading it receives.

TYPES OF NORMALS

Normals come in four varieties: plane normals, point normals, vertex normals, and surface normals. They indicate the orientation (direction) of a point, plane, vertex, or surface curve. If a curve is planar and does not share its points with other primitives, its default point, vertex, and primitive normals are identical, perpendicular to the plane of the curve.

ACTIVATING NORMALS DISPLAY

Activate the Point, Vertex, and Primitive Normal Display in Houdini by enabling the *normals* option in the *View Options* dialog (*Viewport Options Dialog* p. 122), available by clicking the ✛ button at the bottom of the viewport.

Note that the point normals must have been computed first (in a Point SOP, for example). Primitive normals are computed on the spot and only when they are turned on for display. Some systematic primitives, like sphere and cylinder do not have a normal.

CHANGING NORMALS

You can change the normal of an entity in Houdini with SOPs such as the Point and Magnet SOPs. See the *Editing Geometry* section for further information.

6.4 POINT UV'S (TEXTURE COORDINATES)

Normally items are located spatially by XYZ values. To differentiate texture coordinate space from XYZ space, the labels U and V are used instead of X and Y.

In order to place texture maps (images) onto geometry, we must assign texture coordinates to the geometry. A texture map resides in its own (U, V) texture coordinate space. When assigned to the geometry, the (U, V) coordinates designate how to map the image onto the geometry. Texture space should not be mistaken for the parametric space of splines.

Texture coordinates can be visualized in the following manner: Texture maps have their own coordinate space. If the texture were a table cloth with a grid pattern, the color at location 3, 4 on the table cloth remains at location 3, 4 even when the cloth is wrapped around an irregularly shaped object. The color at location 3,4 can be said to be in the table-cloth's coordinate space.

Production Tip: By using a Point SOP, you can swap the position and the texture coordinates. This allows you to model the "texture space". Another Point SOP allows you to swap the position and texture back to their original locations.

6.5 SOME TYPICAL POINT ATTRIBUTES

POINT WEIGHT (W)

This denotes the amount of influence or "pull" a control vertex (CV) has on a Bézier or NURBS curve or surface (if the point happens to be included as a part of one). It should not be confused with the mass of a particle, or the weight of a metaball. See *Splines* p. 238.

DIFFUSE COLOR (CD)

The surface color specified by a triplet of RGB values which range from 0-1.
000 yields black and 111 produces white.

ALPHA

The transparency of a given element, where 1 is fully opaque, and 0 is fully transparent.

VELOCITY (V)

The distance per second the element travels.

7 LIST OF ATTRIBUTES

These are the attributes which are currently reserved for Houdini use. The list may change with no notification or fear of consequence by users who have defined their own attributes. This is a suggested list, not a definitive list.

7.1 ORDER OF ATTRIBUTE PRECEDENCE

The order of precedence for attributes from highest to lowest is:

- Vertex Attributes
- Point Attributes
- Primitive Attributes
- Detail Attributes

Attributes with a higher order of precedence override similar attributes with a lower order of precedence. For more information on Attributes, see *Attributes* p. 233.

7.2 MATERIAL & PHYSICAL ATTRIBUTES

Location can be:

d	Detail attribute
pr	Primitive attribute
pt	Point attribute
v	Vertex attribute

MATERIAL GROUP

Name	Type	Size	Location	Description
mat	index	1	d, pr	Material name specification
rishade	index	1	d, pr	RenderMan shader name
Ca	float	3	d, pr, pt, v	Ambient color (override)
Cd	float	3	d, pr, pt, v	Diffuse color (override)
Cs	float	3	d, pr, pt, v	Specular color (override)
Cr	float	3	d, pr, pt, v	Reflect color (override)
Ct	float	3	d, pr, pt, v	Transmit color (override)
Ce	float	3	d, pr, pt, v	Emission color (override)
Alpha	float	1	d, pr, pt, v	Alpha para/perp (override)
rough	float	1	d, pr, pt, v	Roughness (override)
fresnel	float	1	d, pr, pt, v	Fresnel coeff (override)
apara	float	1	d, pr, pt, v	Alpha parallel (override)
aperp	float	1	d, pr, pt, v	Alpha perp (override)
aroll	float	1	d, pr, pt, v	Alpha rolloff (override)
shadow	float	1	d, pr, pt, v	Shadow intensity (override)
sbias	float	1	d, pr, pt, v	Shadow bias (override)

PHYSICAL GROUP

Name	Type	Size	Location	Description
N	vector	3	pr, pt	Surface Normal
v	vector	3	pt	Velocity
uv	float	3	pt, v	Texture coordinates (UVW)
lod	float	1	d, pr	Level of detail
dir	vector	3	pr	Vector for attractor force
fedge	float	1	pr	Edge force for attractor
fvortex	float	1	pr	Vortex force for attractor
fspiral	float	1	pr	Spiral force for attractor
mass	float	1	pt	Mass
drag	float	1	pt	Coefficient of drag
life	float	2	pt	Life time (index 0=time alive index 1=death time)
id	float	1	pt	Identifying tag
rest	float	3	pt	Rest position

7.3 WHERE ATTRIBUTES CAN BE SET

Following, is a list of where attributes can be adjusted or set:

Attribute Type	SOP to Create	SOP Where it's Used
Point Cd	Point	
Point Alpha	Point	
Primitive Cd	Point	
Primitive Alpha	Point	
Point N	Facet, Point	Point
Point v	Facet	Particle, Spring
Point uv	Texture, Point	
Primitive dir	Attractor	Particle, Spring
Primitive fedge	Attractor	Particle, Spring
Primitive fvortex	Attractor	Particle, Spring
Primitive fspiral	Attractor	Particle, Spring
Point mass	Point	Particle, Spring
Point drag	Point	Particle, Spring
Point life	Particle, Spring	Particle, Spring, Point
Point id	Particle, Spring	

8 SPLINES

8.1 INTRODUCTION

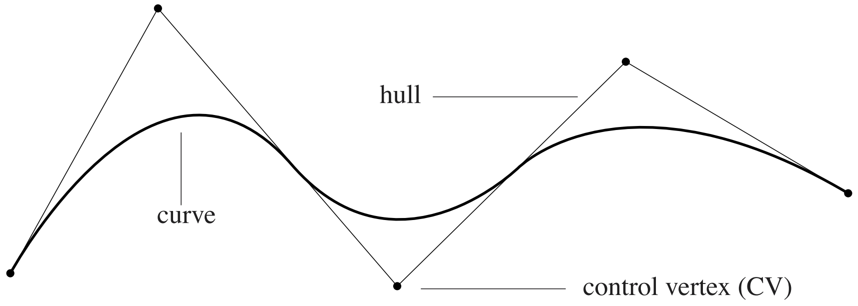
Houdini allows you to create both Bézier and NURBS curves and surfaces (for the purposes of this discussion, however, we'll only be referring to curves). Spline curves and polygons are collectively termed "faces", while grids and spline surfaces are termed "hulls". As opposed to polygonal types, NURBS and Bézier entities are inherently smooth primitives known as *splines*. It isn't necessary to master the mathematics behind what differentiates the two spline types. It is, however, useful to understand some of the concepts that arise from the mathematics of computer-generated curves because they affect your choice of curve type when you start creating in Houdini, and they influence the way you draw that curve.

8.2 LINEAR SPLINES

The most straight-forward way of drawing a curve is by connecting a sequence of points. The resulting curve is a linear spline, and is equivalent to a polygon. There are two major drawbacks to this method of producing a curve. First, in order to produce anything that actually appears curved, you would need a large number of points. Storing and computing all those points is not an efficient use of the computer's resources. Second, manipulating a curve created in this fashion is very cumbersome because, once a point is moved, you lose the smoothness of the shape.

8.3 HIGHER DEGREE SPLINES

The way around the jaggedness produced by linear connectivity is through a series of blending functions. The blending functions, or *bases*, are the mathematical foundation for generating a smooth connection between the control vertices (CVs) of the curve. A spline curve generates a smooth transition between its control vertices by a mathematical blending function that operates on these points. The set of CVs controlling the curve is referred to as the "hull".



8.4 CURVE SEGMENTS

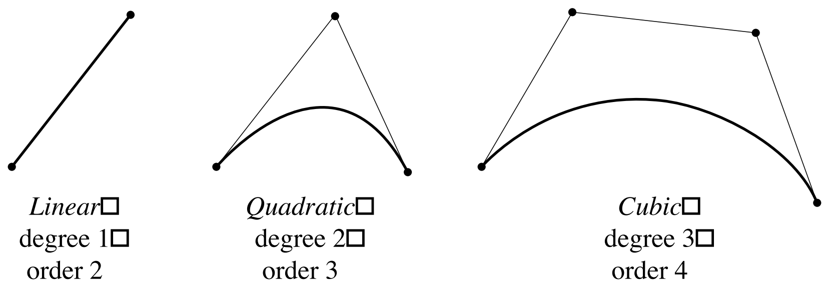
NURBS and Bézier curves in Houdini are piecewise curves made of a number of connected curve segments. The main difference between NURBS and Bézier curves is the level of continuity at the points where the curve segments touch. A NURBS curve will typically be very smooth at these joints (the higher the degree of the blending function, the smoother the connection). Bézier curves have a discontinuity every *degree plus one* points.

8.5 ORDERS

The “degree plus one” formulation is often referred to as the order of the curve. A cubic curve, for example, has a degree of three and, therefore, an order of four.

The degree of the spline in given by the degree of the underlying blending functions. Houdini supports splines whose degrees vary from 1 to 10. The upper bound was chosen for practical reasons and efficiency.

You’ll find that cubic splines are sufficiently smooth and well behaved for most applications. You will seldom need to use other degrees.



From this illustration, we can see that the minimum number of points needed to build a curve equals the order of that curve, unless the curve is closed, in which case only degree CVs will suffice, since the remaining CV is taken to be equivalent to the first CV.

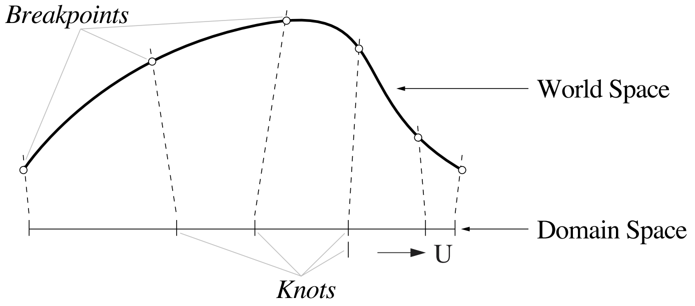
8.6 BREAKPOINTS, KNOTS, AND SPLINE BASIS

The point where curve segments come together is called a breakpoint. It is important to stress that this breakpoint is on the curve itself, not away from the curve like the CVs, which make up the hull.

Breakpoints are images of special values, called “knots”, in what is known as the *parametric space* or the *domain* of the spline. The domain, which is simply a sequence of knots in ascending order, together with the spline order and the spline type define a spline *basis*.

Imagine the domain of a curve as a segment going from zero to one or (for example) -12.7 to 83.2, whose size and origin are given by the values of its two end-knots.

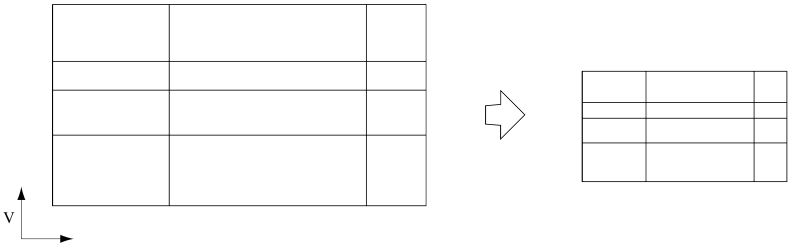
Similarly, a surface is defined by two knot sequences forming a rectangular (U,V) domain. The knot sequences must always be sorted in ascending order.



Relation of World Space and Domain Space

Since knots are the ingredients of the domain, they divide a curve's domain segment and a surface's domain rectangle into smaller pieces whose size relative to each other is often more important than the total size of the domain. Similarly, in world space, the areas delimited by breakpoints divide a curve into curve segments and a surface into patches.

Depending on the type of spline, the relative knot distances usually determine the shape of the spline given a fixed set of control vertices. The size and the origin of the domain are relevant when identifying a surface's texture space with its parametric space. Then, if the texture is expected to cover the entire surface only once, the domain of the surface must be a unit square. Mapping a domain to a new range and origin does not affect the shape of the spline primitive because the knot ratios are preserved.



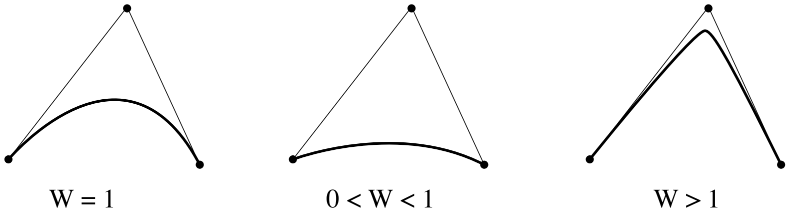
Ratios between knots are maintained during transformation

The knots need not be evenly spaced in the domain. The more knots there are in one area, the smaller the spline segments and, therefore, you have a greater degree of control over the spline in that area. If several knots are placed at one value, something called a *multiplicity* is produced. Not all spline types allow multiplicities to occur.

8.7 RATIONAL SPLINES

Houdini supports two types of rational splines: NURBS, and Bézier. Each CV of the curve has X, Y, and Z coordinates that determine its position in world space. There is also a fourth component for each CV called *W*. The *W* component determines a CV's weight (see also: *Point Weight (W)* p. 235). The weight determines the "pull" (like a magnet) of a CV on the spline curve. The value of the *W* component makes a spline rational or non-rational. A non-rational spline has only equal weights (typically, *W=1*), while a rational spline contains at least one different weight. While non-positive weights (where *W* is less than or equal to zero) make sense in theory, they tend to generate unintuitive shapes and cause the spline to break away from its convex hull. For practical reasons, Houdini supports only positive weights (W > 0).

The higher the weight of a CV, the sharper the spline around that CV. For large weight values, the spline will almost go through the CV. Similarly, weights smaller than one tend to flatten the spline in the area influenced by that CV.



Effect of CV weight

However, it isn't simply the size of the weight that causes a fluctuation in sharpness. An equally, if not more, important element is the relative difference between weights. The more equal the neighbouring weights, the smaller their influence over the given region and, consequently, the less rational the spline. For example, if all the weights of a spline curve are one thousand, the shape of the curve will be identical to a non-rational curve.

In Houdini, certain models, like the perfect NURBS circles, are normally built rationally. Although you can create rational models yourself in the modeller and elsewhere, we recommend that you use weights sparingly because they increase the complexity of the model (which may result in decreased system performance) and they may also lose their effectiveness when applied to neighbouring curve regions.

Of the spline types supported in Houdini, NURBS curves give you a greater degree of control over local portions of the curve and over its smoothness.

8.8 NURBS CURVES AND SURFACES

NURBS is an acronym for Non-Uniform Rational B-Spline. A NURBS curve employs a series of blending functions called "bases" to generate a smooth curve from a sequence of control vertices (CV's) which define a NURBS hull.

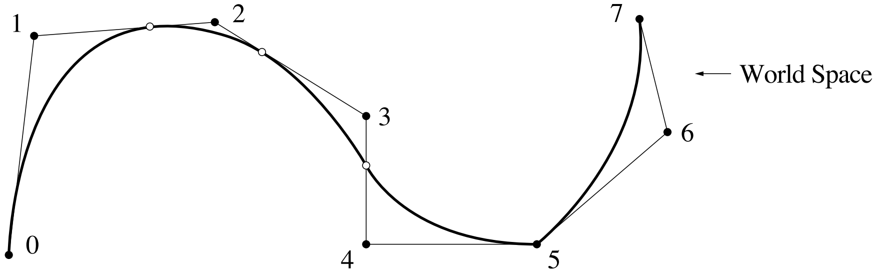
The primary advantage of using a NURBS curve is that moving a CV only affects a local portion of the spline while also maintaining the continuity of the curve, even at its breakpoints. This allows you to "pull and tug" on the CVs of the NURBS curve or surface to generate a desired shape without causing kinks or discontinuities.

The shape of a NURBS curve is greatly influenced by the relative distances between its knots. The knots appear in ascending order, and are possibly repeated. A repeated knot is said to have a *multiplicity.*

In a Bézier curve, all knots are unique and, therefore, multiplicities aren't produced. The parallel between Bézier and NURBS knots is that a Bézier knot is similar to a

NURBS knot with maximum multiplicity. The Bézier discontinuities mentioned earlier happen at these knots. Similarly, a NURBS curve will have a discontinuity where a knot is at maximum multiplicity. Maximum multiplicity occurs when a knot is repeated *degree* times in a NURBS basis. Both NURBS and Bézier curves will have a CV on the curve at the point of discontinuity.

If the multiplicity happens at the end of the curve, the NURBS curve is considered "clamped." Typically, NURBS curves are clamped at both ends but closed curves are usually unclamped; Bézier curves are always clamped.



A clamped Quadratic (degree 2) curve on☐
U = [0, 0, 0, 1/6, 2/6, 3/6, 4/6, 4/6, 1, 1, 1]☐ ← Domain Space
with a cusp at u = 4/6, because 4/6 is repeated *degree* times

From this, it follows that the shape of a NURBS curve, given a set of CVs, is determined by the relative distance between knots. Typically, there are two types of knot parameterizations: uniform and chord length. In the first, knots are spaced evenly. In the second, the distances between knots are determined by the distances between successive CVs. Uniform parameterization is recommended for regular shapes while chord length is used for free-form shapes. A third type of parameterization, called "centripetal", is similar to chord length and is best suited for sharp curves.

CREATING A SHARP POINT IN A NURBS CURVE

It is sometimes desirable to simulate a discontinuity (a sharp corner point) along a NURBS curve. This can be done one of three ways:

i) Change the weight of a selected CV via the *Curve > Parameters* dialog in the Model Editor to something high like 10,000. This gives the CV so much "pull" that it draws the curve almost right through it.

ii) If you drag the two adjacent CVs of a cubic curve onto a middle CV, it will look like a sharp corner point. When this is done, it is called raising the *Multiplicity* of the CV. Maximum CV multiplicity occurs when adjacent "degree" CVs overlap.

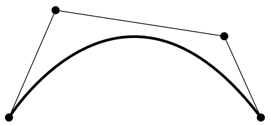
iii) Make "degree" knots identical. When this is done, it is called raising the multiplicity of the knot. You can do this in the Refine SOP by choosing the *Subdivision* option, or in the Model Editor by selecting the Refine state and dividing with the middle mouse button (🖱).

CLAMPED AND UNCLAMPED NURBS CURVES

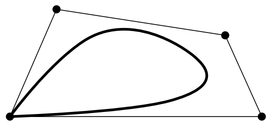
A clamped curve touches its endpoints. An unclamped curve doesn't. Much of the time you will work with clamped curves. The unclamped case is generally useful for closed curves.

In Houdini, you can build closed NURBS curves, and these can be clamped or unclamped (see below). A closed, clamped curve will show a discontinuity where the two end points touch.

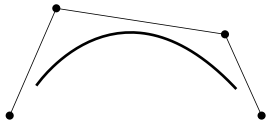
There are four types of end point conditions for NURBS curves:



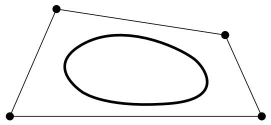
Open (cubic) NURBS
clamped



Closed (cubic) NURBS
clamped



Open (cubic) NURBS
unclamped



Closed (cubic) NURBS
unclamped

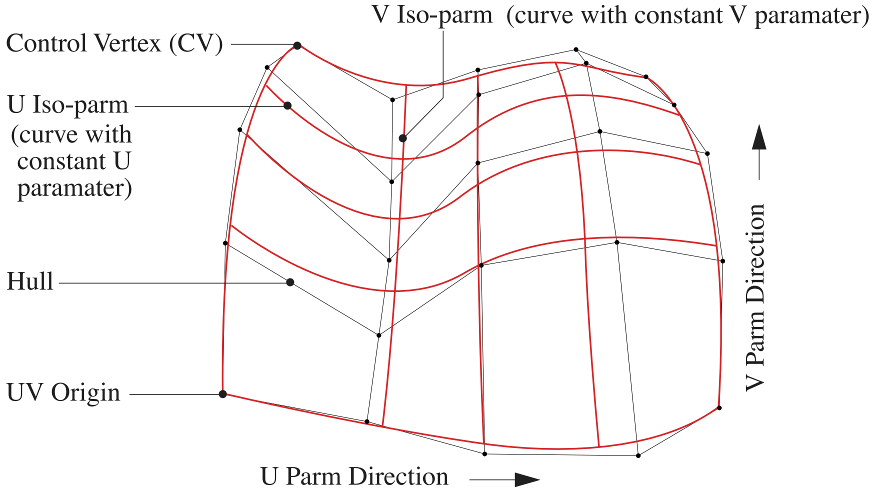
You can experiment with these properties–both for curves and for surfaces in the Curve/Hull page of the Primitive SOP.

PERIODIC CURVES

A closed unclamped curve will wrap around itself, and is known as a periodic curve because the knots of the wrapped portion are actually a cyclical repetition of the original knots. When you save a periodic NURBS curve to a file, the periodic knots are not saved, but are generated automatically upon loading the file.

8.9 NURBS SURFACES

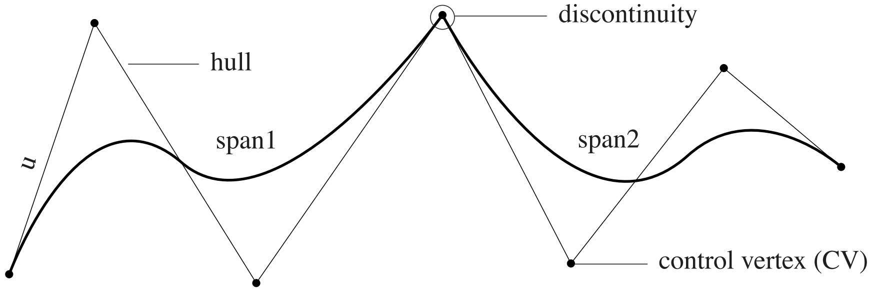
A NURBS surface has a topology similar to that of a mesh primitive (see *Meshes* p. 226.) In a NURBS surface, each node of the UV coordinate matrix represents a CV connected by rows and columns to form a NURBS hull. This allows modelling of complex smooth surfaces, whose shape is changeable simply by moving the CVs.


V Iso-parm (curve with constant V paramater)
Control Vertex (CV)
U Iso-parm (curve with constant U paramater)
Hull
UV Origin
V Parm Direction
U Parm Direction

Tip: To create an open cubic NURBS surface, you need at least 4 x 4 CVs. In general, for an open surface with U Order *m*, and V Order *n*, you need *m* x *n* CVs. For an open NURBS curve of order *m*, you will need at least *m* CVs to define it properly.

8.10 BÉZIER CURVES AND SURFACES

Béziers are similar to NURBS, however, they always touch the end-points of the hull (are clamped) and possibly CVs in between (depending on the total number of CVs and the order of the curve). The main difference between Béziers and NURBS is that Béziers have a discontinuity (which might look like a sharp point in the curve) at regular intervals based on the order of the curve.



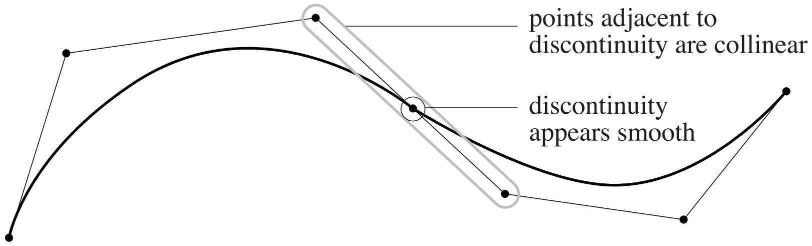
In the Model Editor, you set the order of the curve via the *Curve State* > *Parameters* dialog (*Orders* p. 239).

For example, if the order of the curve is 4, then the degree will equal 3, meaning you will have four CVs in each curve span. Between each span there will be a discontinuity. Order four curves are best known as “cubics”.

Whereas NURBS curves change only a local portion of the curve when a control point is moved, the entire span within a Bézier curve is affected when a control point belonging to that span is moved.

If we set the order of the curve to a low number like 2, then each span of the Bézier will be between one CV, and it will look like straight lines. Therefore setting the order to 2 results in a curve that looks like a polygon (composed of straight lines). The curve is called "linear".

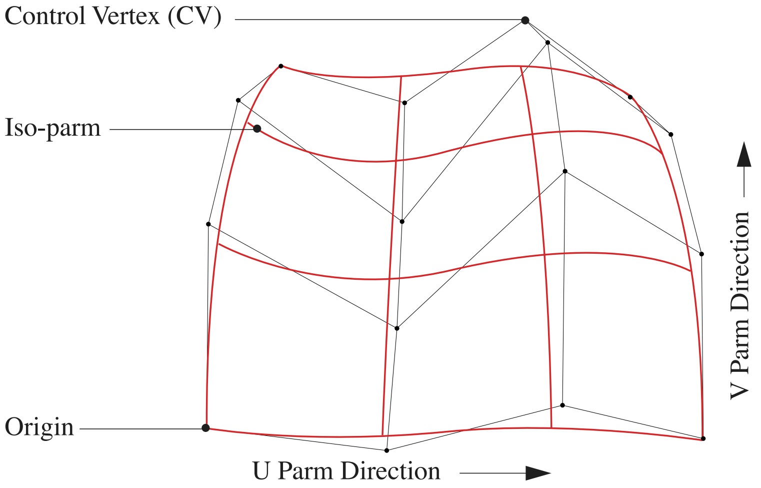
The only way to avoid the "sharp corner" of a discontinuity in a Bézier curve is to make the CVs adjacent to the point of discontinuity collinear as illustrated below.


points adjacent to discontinuity are collinear
discontinuity appears smooth

In the Model Editor, a smooth Bézier curve can be built using the *Breakpoints* option in the Curve state.

8.11 BÉZIER SURFACES

A Bézier surface uses a topology similar to that of a mesh. In a Bézier surface, each node of the UV coordinate matrix represents a CV connected by rows and columns of Bézier hulls. This allows modelling of complex smooth surfaces, whose shape is changeable simply by moving the CVs.



Note: Open Bézier surfaces can only be built with the number of UVs equal to *multiples of the order-1*, plus one. For example, for an order 4 curve the number of Us or Vs can equal 4, 7, 10, 13, 16, etc. because if the order is 4, then we take the *order-1* (=3) and multiply by the number of spans + 1 as in the following: 4=(1*3)+1, 7=(2*3)+1, 10=(3*3)+1, 13=(4*3)+1, and 16=(5*3)+1. If the Bézier surface is wrapped in U and/or V, the formula above loses the "+1"

8.12 CURVES ON SURFACES ("PROFILE" OR "TRIM" CURVES)

Houdini supports curves on surfaces (also known as *trim curves* and *profile curves* or *profiles* for short), which are defined in and bound by the size of the parametric space (i.e. domain) of their parent surface.

TYPES OF PROFILES

There are three types of profiles in Houdini: polygons, NURBS curves, and Bézier curves. The definition of each profile type is very similar to that of its 3D equivalent. For example, a NURBS profile can be open or closed, clamped or unclamped, rational or non-rational, and its order can vary between 2 and 11. The fundamental difference between profiles and 3D faces is that the profiles are 2D curves whose CVs are (U,V,W) points in the domain of the parent surface; CVs of 3D faces are X,Y,Z,W quadruplets in object space. For both types of faces W is the rational component, (W=1 is the non-rational case).

VISUAL PROPERTIES

The visual representation of a profile, then, is a curve that is always glued to the surface. The 3D shape that we see is merely an image of the 2D geometry that makes up the profile, on the surface. This means that, whenever the surface changes shape, the profile image on the surface follows it. It also means that no part of the profile can ever be lifted off the surface.

If the entire profile or part of it goes outside the boundaries of the surface domain, it becomes invisible because the surface itself is not defined past those limits. An invisible profile is still valid and can be brought back into view. See the Profile and Primitive SOPs for ways to manipulate a profile.

A common use of the profile curve is for trimming, which keeps or removes the part of the surface enclosed by the profile (see the *Trim OP* p. 819 in the *Editing Geometry* section). This is why profile curves are sometimes also called trim curves.

PROFILE NUMBERING AND DISPLAY

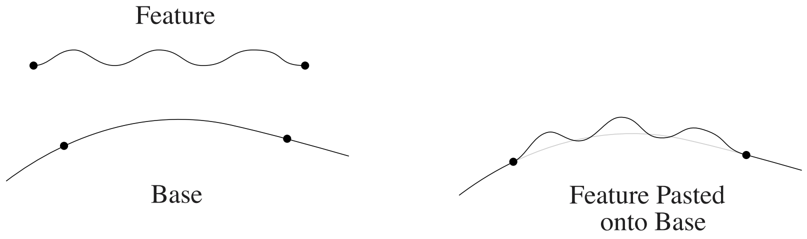
Profiles are displayed in Houdini as dotted curves. Their numbers can be displayed using the Display Options dialog, by clicking on the "Profile Number" icons in the *Accessories* page. A profile number is always prefixed by the primitive number of its parent surface. For example, "0.2" is the third profile of first primitive in the geo detail (numbering starts at zero).

A profile can co-exist with 3D primitives in primitive groups. A resulting mixed group would specify its elements like this: "1 0.2 6 2.9-2.17 4.*" .

8.13 PASTING

Pasting is a process which allows you to take to two or more 3-D surfaces (called “features”), and add their effect on top of another 3-D surface (called the “base”). This is done without increasing the complexity of the base surface, yet allows the added detail to move freely on the base after being pasted. This is useful for such things as maintaining a library of facial features and pasting them onto the 3-D model of a person’s head.

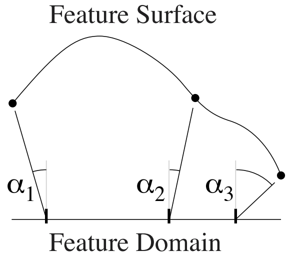
In the process of pasting, the 3-D surface moulds itself onto the shape of the underlying base surface – in effect, the feature’s UV space is placed within the base’s UV space.

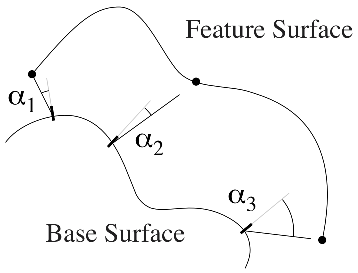


This process is maintained in a hierarchical fashion, thus allowing multiple features to be pasted (e.g. nose, ears, mouth) onto a single base. Features may also overlap. More features may then be pasted onto existing features.

HOW IS THE PASTE CALCULATED?

The angles formed by mapping the domain of the feature to the surface of the feature are computed . Then, these angles are added to the normals of the base shape in order to derive the XYZ locations for points on the pasted surface.

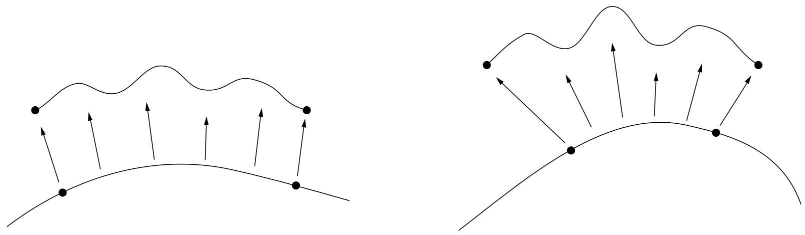




SOME NOTES ON PASTING QUALITY

The vectors which connect the feature domain and the feature surface are called “tentacles’. The length, angles, and the number of tentacles determine the shape and the quality of the paste.

Thus, if the base has a high amount of curvature and the tentacles diverge inward or outwards, greater distortion in the paste will result.



From this fact, we can make several observations which will help us in regards to decisions that affect pasting quality:

- 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.

SPAWNING

It is also possible to add detail to a base by extracting a pasted sub-surface from it. The resulting surface will share the shape and the underlying domain of the base in that area. This procedure is called "spawning". For more information, see *Growing Detail from Within (Spawning)* p. 249, and the *Paste OP* p. 661 in the *Geometry* section.

PASTE HIERARCHIES

A hierarchy of pasted splines is called a "paste hierarchy" or "muli-resolution surface", and is represented as a primitive in the geo-detail. It's number is displayed in brackets (e.g. (5)) when primitive numbers are enabled in the Viewport options. The brackets should not be used when specifying the primitive number.

A spline surface can belong to only one hierarchy at a time.

CURRENT LIMITATIONS

- You cannot paste across multiple paste hierarchies.
- Most, but not all SOPs support the "Pasted" primitive type.
  Those that do not will either ignore the hierarchical primitive or delete it.

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 (parametric and projective)

This is done using the Paste SOP, with two inputs: the feature and the base.

- Parametric Paste
- Along Vector (Projective) Paste

1. In a Parametric Paste, the Paste SOP 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 (spawning)

If you use only one input to the Paste SOP, 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.

8.14 UNPASTING

The deletion of surfaces is available in the Delete and Unpaste SOPs, as well as in the Model Editor.

The Unpaste SOP 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.

By preserving the hierarchical structure of the unpasted surfaces, the resulting sub-hierarchy can be re-applied 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 SOP. One case of repasting is worth mentioning for its practical use.

Assume you have used the Paste SOP 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 SOP 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 array of tools.
- Repaste the feature on the same base surface using the "As Is" page with *Shape Preservation* enabled.

9 HOW FORCE FIELDS WORK

9.1 INTRODUCTION

Force fields can be used to affect the Particle or Spring SOP. The force field is defined by one or more metaball primitives that have been fed into the Force SOP. The attractor output is then fed into the third (right-most) input of the Particle or Spring SOP.

A metaball defines a field of effect. The meta-surface is the point in the field where the value of the field is 1. Thus when two metaballs are merged, their fields get added. When metaballs are used for attractors, the fields are also added together.

9.2 TYPES OF FORCES

When a single metaball is used as an attractor, there are four types of forces which can be added to the force field. These are:

radial force	A force pulling particles to the centre of the force field. If the force is negative, the particles will be repelled from the centre.
axial force	A force which pulls particles along the axis specified.
vortex force	A tangential force which causes the particles to spiral around the axis specified.
spiral force	A force which pulls particles toward the axis specified. If the force is negative, the particles will be repulsed from the axis.

9.3 OTHER NOTES

The axis specified is in relation to the space of the metaball. Thus, if the metaball is rotated by using a Transform SOP, the axis specified will also be rotated. This allows for different metaballs to push particles in different directions.

When multiple metaballs are merged together as attractors, each metaball acts independently on the particle in question. However, the forces of all metaballs are cumulative, causing the particle to be affected by all the different attractors.

Outside the hull of a metaball (visible as a guide geometry in the Particle SOP, or by turning on display of hulls in the viewport), the attractor will have no effect. At the centre of the metaball, the scale of the forces will be whatever the weight of the metaball is set to. By adjusting the weight and kernel type of each metaball, different force field types can be generated.

10 CONVERTING BETWEEN GEOMETRY TYPES

Houdini handles many geometry types, each with their own uses and strengths. It is important to be able to convert between them, as different types may be required for a given situation: creation, editing, and rendering. Houdini has the ability to convert between different geometry types at any point. This allows a great deal of flexibility in the modelling process.

For example, say we are designing a couch for a set in a computer game, and the game developers require texture mapped quadratic polygons (quads) as the final output. It may be best to create the hulls for the couch using NURBS or Bézier curves because of their ability to easily handle organic curves. We can then convert these to polygon meshes for a certain type of surface deformation. Finally, we can convert the polygon meshes into quad polygons for texture mapping and output. Alternately, we can do all the work with NURBS

This is one example of mixing geometry in a linear manner. Modelling with different geometries is allowed and encouraged. To convert between different geometries, use the Convert SOP (see *Convert OP* p. 512 in the *Geometry* section).

When converting a spline surface containing profile curves to a mesh or set of polygons, the profile information is lost.