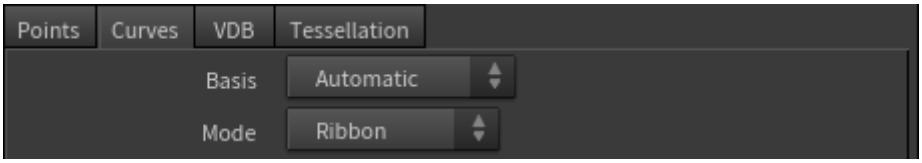
Curves

Rollover image

It is possible to render curves with Arnold. An Arnold *Curves* tab is provided.

> Currently, there is no checkbox to avoid SSS computations in point and curve primitives. However, the use of SSS is not supported with point and curve primitives and is therefore not recommended.



Basis

Describes how the curve is formed from the control points. The available basis choices are Bezier, B-Spline, Catmull-Rom, or Linear:

bezier: the curve touches the first and last points, and then touches each interior and shared fourth point. That means a curve with 16 points will have the curve visually touch the 1st, 4th, 7th, 10th, 13th, and 16th points, where the 4th, 7th, 10th, and 13th points are shared between curve segments. Bezier curves thus form a new segment every 4th point, with the 4th point forming the first point of the next curve segment. Or more generally, Bezier curves have (npoints - 1)/3 segments.

b-spline: the curve likely does not touch any of the control points, unless you repeat a control point three times. These curves have 3 fewer segments than control points, where the segments are evenly distributed over the curve.

catmull-rom: the curve does not touch the first and last control points unless you repeat a control point three times, but it does cause the curve to pass through the other control points and swing wide if needed to do so. It has the same number of segments as b-spline, or three fewer segments than control points.

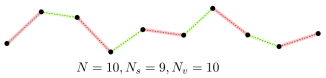
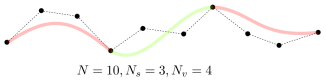
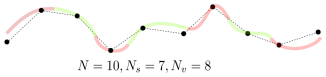
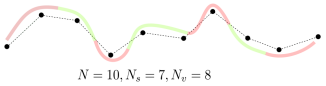
linear: the curve touches each point, and is a straight line in between points. Each point and its next point forms a segment.

As in other shapes, constant data has a single value for the entire curves shape. Uniform data has a single value per curve. But varying data for curves requires one value at the start of each curve segment, and one at the end, so that if a curve has 10 segments it will have 9 shared values, and then one value at the beginning and one at the end of the curve for a total of 11 values. For a linear basis, this is straightforward, as there is a single value per control point.

For Bezier basis this gets more complicated: each segment spans 4 points, with the fourth point shared with the next segment's first point, so a Bezier curve with four points has one segment and thus two varying values; a Bezier curve with 7 points has two segments and three varying values; a Bezier curve with 10 points has three segments and four varying values, and so on.

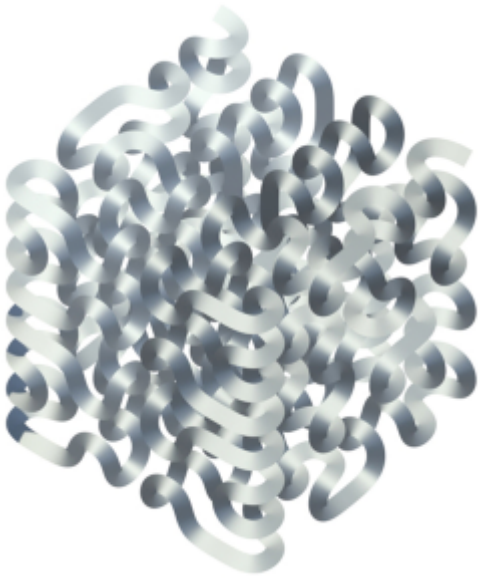
B-spline and Catmull-Rom are both the same; there are npoints-3 segments, and so there are npoints-2 varying values.

As described above, the effective number of curve segments for a given number of control points depends on the basis used, which is summarized in the table below. Note that the number of varying values (e.g. radii) is always equal to the number of segments plus one.

Basis	Number of segments $N_s(N)$	Number of points $N(N_s)$	Number of varying $N_v = N_s + 1$	Schematic (point, segment)
Linear	$N - 1$	$N_s + 1$	N	 $N = 10, N_s = 9, N_v = 10$
Bézier	$\frac{N-1}{3}$	$4 + 3(N_s - 1)$	$\frac{N+2}{3}$	 $N = 10, N_s = 3, N_v = 4$
Catmull-Rom	$N - 3$	$N_s + 3$	$N - 2$	 $N = 10, N_s = 7, N_v = 8$
B-spline	$N - 3$	$N_s + 3$	$N - 2$	 $N = 10, N_s = 7, N_v = 8$

Mode

There are three algorithms for rendering curves in Arnold.



Ribbon

Thick

Ribbon

Ribbon mode is recommended for fine geometry such as realistic hair, fur or fields of grass. These curves are rendered as camera-facing flat ribbons. For secondary and shadow rays, they face the incoming ray direction. This mode doesn't look so good for very wide hairs or dramatic zoom-ins because of the flat appearance. This mode works best with a proper hair shader (perhaps based on a Kay-Kajiya or Marschner specular model).

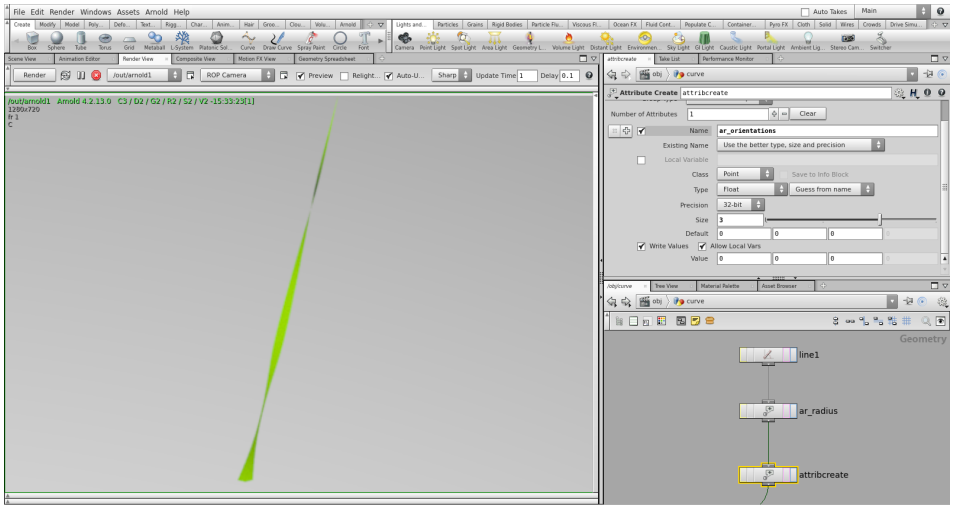
Thick

Thick mode resembles spaghetti. It has a circular cross-section and a normal vector that varies across the width of the hair. Thick hairs look great when zoomed in, and are especially useful for effects work, but their varying normals make them more difficult to antialias when they are small. You can use any shader with this rendering mode, including lambert, phong, etc.

Oriented

Oriented mode allows the curves to face in a given direction at each point. This is more useful for modeling blades of grass, tape, and so on.

You can modify the orientation by using the name ***ar_orientations*** with an **attribcreate** node (*Tab > Attribute > AttribCreate*). The ar_orientations attribute is per point and is a vector representing the direction towards which the curve ribbon is facing at this point of the curve.

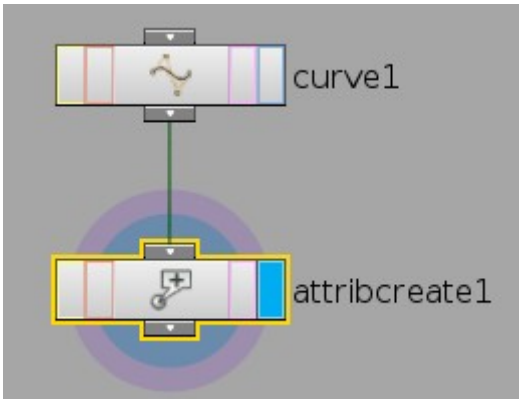


Attribcreate name set to 'ar_orientations'

Overriding Curve Width

The **Default Radius** property can be overridden with one of the following attributes to provide more control or randomization of the width.

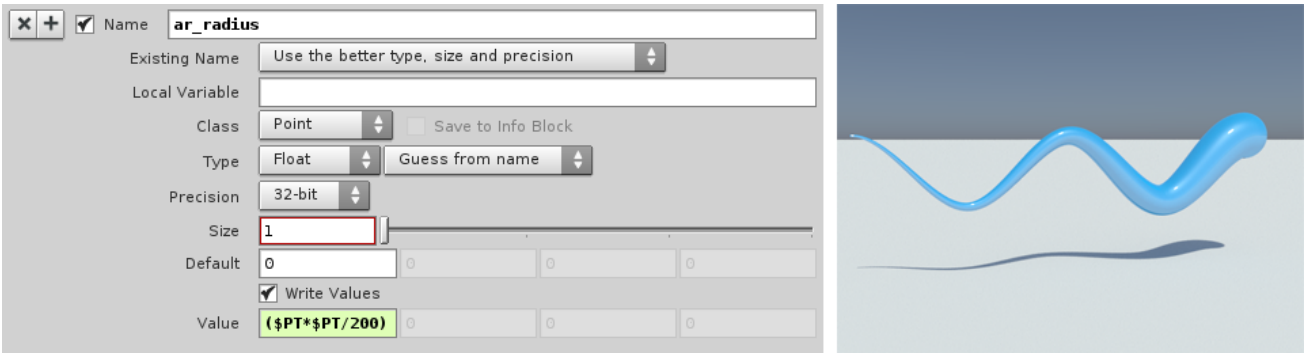
To implement this, use an attribcreate node (*Tab > Attribute > AttribCreate*) to modify the attribute.



HtoA will check for these three attributes, in order:

- ar_radius (point float, arnold-specific)
- pscale (point float, traditional Houdini attribute)
- width (vertex float, from Houdini hair setup)

Note that Arnold specifies a radius for curves but the traditional Houdini **width** and **pscale** attributes are diameters. When using width or pscale, HtoA will half the value entered to use as the radius.



The curve width is overridden using the ar_radius attribute with the value ($PT*$PT)/200