Curves

Introduction

Curves can be used to render large numbers of thin strands of geometry very efficiently, making this primitive particularly well-suited for modeling fur, hair, and blades of grass. This geometry type is an approximate representation, designed to render more efficiently in both time and memory compared to an equivalent surface of circular cross section. The trade-off is that when viewing curve geometry up close, there can sometimes be geometric artifacts; these artifacts tend to not be perceptible as the curves become more distant from the camera. RenderMan supports both flat (ribbon-like) and round (tube-like) curves, and it is possible to choose from a variety of basis functions.

Parameters

Width

The width of a curve can vary over the length of the curve. It is possible to specify a constant width that applies to all curve strands in the primitive, or widths can be specified to be different on a per-curve strand, per-segment, or per-vertex basis.

Constant Width (constant width of 0.005 along length of each curve)

Varying Width (tapered width ranges from 0.005 to 0.0 along the length of each curve)

When specified per-segment or per-vertex, negative values for width can be used along curves: as the value of width is interpolated along the curve, the curve will taper to a sharp tip and the portion of curve with a negative width will disappear. This can be used to slightly randomize the length of hair in a clump without changing control points or their animation/simulation.

Basis

Given a set of control points that the user supplies as vertices, which composes the hull of the curve, the basis function determines how the actual rendered curve appears. For example, some basis functions, such as the Catmull-Rom basis, guarantee that the rendered curve will pass through all interior, non-endpoint control vertices, whereas others, such as B-spline, typically do not pass exactly through any of the control vertices. Each basis function has different trade-offs in terms of smoothness, controllability and ease-of-use.

- **Bezier** - A Bezier basis generates a 3D curve that passes through every third control vertex. The remaining control vertex positions are used to determine the incoming and outgoing tangent of the curve of the adjacent vertex that the curve does pass through. When using a Bezier basis with curves that contain multiple segments (that is, more than 4 control points), some care should be taken to ensure that incoming and outcoming tangents line up at vertices that are on the rendered curve, or there will be a discontinuity in the smoothness of the curve where it suddenly changes direction.
- **B-Spline** - While a B-Spline basis function can be somewhat less intuitive to use in that it generates a curve that typically only approximately comes close to passing through the given control vertices, an advantage is that this basis function tends to yield very smooth curves. Discontinuities in the smoothness of the curve are possible if multiple control vertices are repeated consecutively (aka multiplicity).
- **Catmull-Rom** - This basis function generates a curve that is guaranteed to pass through every interior, non-endpoint control vertex, and it tends to generate smooth curves, although there can sometimes be unexpected "wobbles" in areas of high curvature with this basis function choice (see images below for an example). Using a B-Spline basis instead can smooth out the wobbles, although a B-Spline basis typically produces a curve that does not pass through the control vertices exactly, unlike the Catmull-Rom basis (for non-endpoint control vertices).
- **Linear** - Specifying linear curves yields straight line segments, continuous linear segments in a curve are joined with a round cap.

The example below shows how different basis affects the shape of a curve given the exact same control vertices: while control vertices and the curve hull is dawn in white, the curve is drawn in yellow.

B-Spline
Bezier
Catmull-Rom
Linear

It is common practice to use B-Spline and Catmull-Rom curves with multiplicity at curves extremities to better control where the curve begins and ends. Note that in order for such types of curve basis to end at the exact position of the first and last curve control vertex, these have to be repeated one more time Catmull-Rom and two more times for B-Splines.

B-Spline with multiplicity
Catmull-Rom with multiplicity

Mode

RenderMan supports both flat (ribbon-like) curves and round (tube-like) curves, demonstrated in the images below.

- **Flat Curves** - usually recommended for blades of grass or other geometry where the curve geometry is relatively more "ribbon-like" than tube-like in appearance. User-normals are specified for flat curves that will twist around so that their orientation always follows the user-specified normals (for both camera and other ray types). No-normal curves (always camera facing) are not supported. Flat curves are no appropriate for foliage or other "flat" surfaces where the geometry is roughly rectangular and the desired silhouette is obtained with a cut-out texture. For these it is much better end more efficient to use polygons instead.
- **Round Curves** - typically recommended for hair and fur since this usually represents the underlying hair geometry with greater fidelity (although not in all cases), and as a direct consequence round curves do tend to produce a more accurate reproduction of occlusion and shadowing for fur and hair compared to flat curves. User normals must not be present when using round curves; if user-specified normals are provided, they will take precedence over the round curve default and as a result the curve will be rendered as a twisting ribbon-like curve that follows the user-specified normals (e.g., a flat curve). Currently, round curves tend to exhibit better time and memory performance compared to flat curves. It is highly dependent on the BxDF and shading model, but in some cases, the tube-like varying normal produced when using round curves can yield more interesting shading results.

Flat Curves with User-Specified Normals
Round Curves

Opacity

Like all geometry in RenderMan, if the BxDF attached to a curves primitive computes presence or opacity, this will control whether the curves are present on screen and to what percent. To improve raytracing performance, presence and opacity signals are always cached for curves. There is also an optimization implemented for round curves where if the opacity is specified in the "Os" primitive variable, then a more efficient code path is taken when computing shadowing which avoids the need to run any BxDFs or lookup into the opacity cache.

Minimum Width

An option can be used to specify the minimum width for curves. When specified, this artificially widens curves so that they are at least as wide as the specified minimum width (measured in pixels), with a corresponding decrease in presence. The effect is that the curve looks about the same when rendered, except that the high spatial frequency geometry is now band-limited thus minimizing aliasing artifacts, which just means that fewer camera samples are needed in order to anti-alias the curve (and thus fewer rays need to be traced, resulting in faster performance). Useful values for minimum hair width range from 0.0 to 1.0.

Option "curve" "float minwidth" [0] (default: 0)

curve:minwidth = 1.0 with 64 samples per pixel

curve:minwidth = 0.2 with 64 samples per pixel

Notice in the example images above that the render with curve:minwidth=1.0 has converged after 2.5 minutes using only 64 samples-per-pixel, whereas visual inspection reveals that the render with curve:minwidth=0.2 (also rendered using 64 samples-per-pixel) still has some areas with not-fully-converged aliasing artifacts after the same 2.5 minutes of rendering (which do ultimately resolve with more samples). Minwidth is not a physical-based control, you may notice the curve:minwidth=1.0 setting has introduced a perceptible amount of bias in the render (that is, it is slightly smoother in a few areas), although the introduced bias is not visually objectionable, in some cases it considered desirable as it produces a softer look.

Best Practices

When rendering hair or fur, it is recommended that:

- Round curves should typically be enabled when appropriate. Doing so will ensure high quality hair shadowing in any lighting configuration by intersecting against cylindrical hair geometry instead of flat ribbons, resulting in significantly more realistic occlusion.
- Many hair or fur strands should be packed into relatively few Curves primitives (e.g., 10,000 or more strands per Curves primitive), rather than rendering millions of Curves primitives that each are composed of just one or a few strands of hair.
- If a model is composed of several curves objects that substantially overlap in volume (e.g. fur coat and undercoat), the best performance is achieved when these objects shares the same values for transformation, visibility flags and group membership.