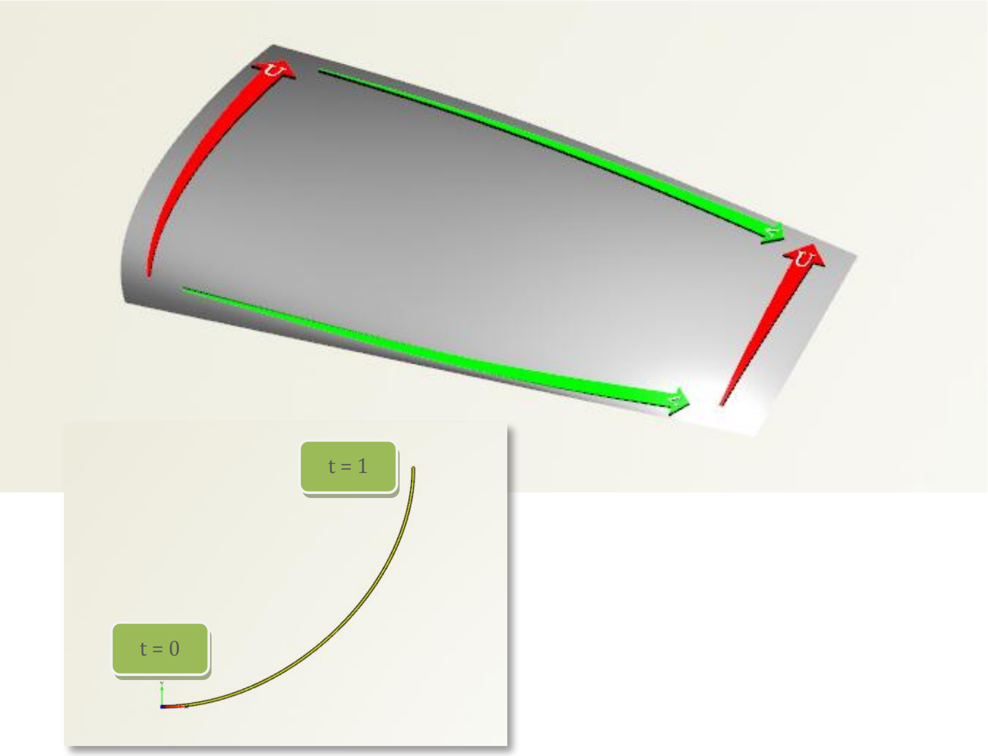
Overview

Introduction

In CAESES, the mathematical definitions of curves and surfaces are important. For instance, curves always have a certain orientation, i.e. a starting and a terminating position. The orientation is then relevant when generating surfaces from curves or when creating sub surfaces.

This brief overview gives you a first idea of how curves and surfaces are defined. In addition, all curves and surfaces provide a set of useful functions that can be accessed and utilized. Some of these functions are presented.

For this overview, it is recommended to first check the previous tutorials of the "Getting Started" section.

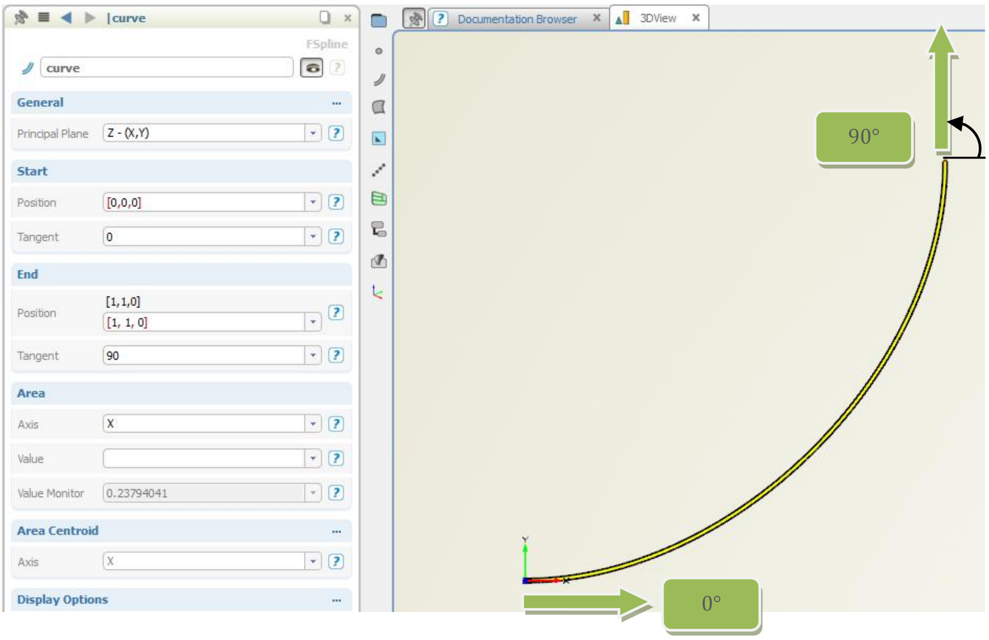


1

Example Curve

As an example, let’s create a simple *f-spline* curve (note that all descriptions also apply for any type of curve, not just for *f-spline* curves):

- Create an *f-spline curve* via *CAD > curves > f-spline curve.*
  Keep the default start and end positions.
- Set the start tangent to “0” degree.
- Set the end tangent to “90” degree.


|curve

FSpline

curve

General

Principal Plane	Z - (X,Y)

Start

Position	[0,0,0]
Tangent	0

End

Position	[1,1,0] [1, 1, 0]
Tangent	90

Area

Axis	X
Value	
Value Monitor	0.23794041

Area Centroid

Axis	X

Display Options

✓ Simple 3D-positions with xyz-coordinates can be set by using squared brackets. For instance, the end position of the *f-spline* curve is given by “[1,1,0]” (x=1, y=1, z=0).

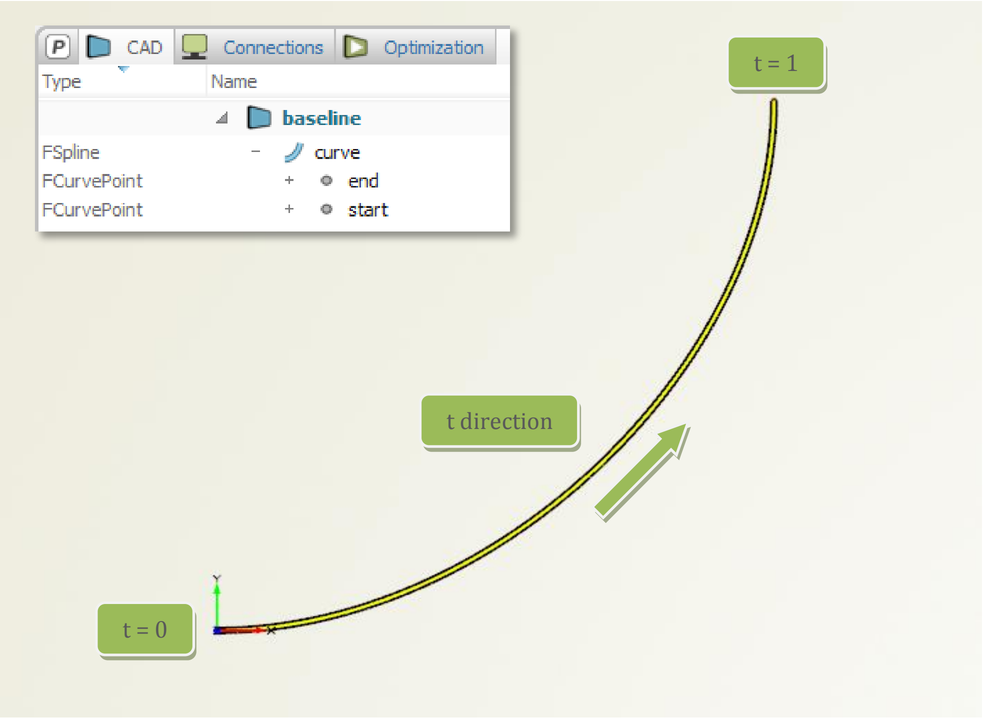
✓ For the *f-spline* curve, the angle settings are related to the specified principal plane which is the xy-plane by default, see the editor attribute “Principal Plane” showing “Z – (X,Y)”.

Using the Right-Hand-Rule, the 0° position is along the x-axis. In the screenshot above, angles are measured in a counter-clockwise direction (mathematically positive) from x- to y-axis.

The orientation of the curve (indicated by the green arrows) is important for the angle settings.

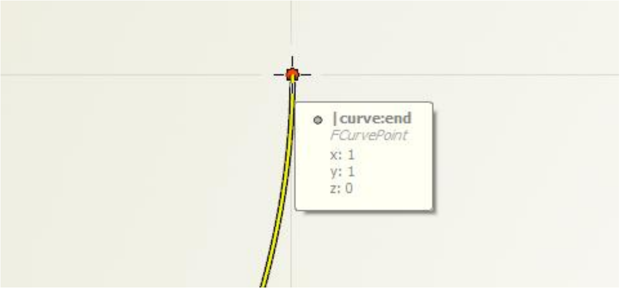
2 Curve Basics

Curves are defined by means of parameter "t" which always runs from "0" (start) to "1" (end). The start and end positions are directly available in the *CAD* tree right below the curve object (expand node). You can also address or utilize them via "curve:start" and "curve:end" ("curve" stands for any user-defined name, of course).



Type	Name
	baseline
FSpline	– curve
FCurvePoint	+ end
FCurvePoint	+ start

These positions are also highlighted if you move your mouse cursor over the terminating ends of the curve in the 3D view. From there, they can be chosen using drag & drop: For instance, click on the end position "curve:end" in the 3D view, keep the left mouse button pressed and drag the position into another editor (where a position is expected, e.g. the start position of a *line* or another *f-spline* etc.).



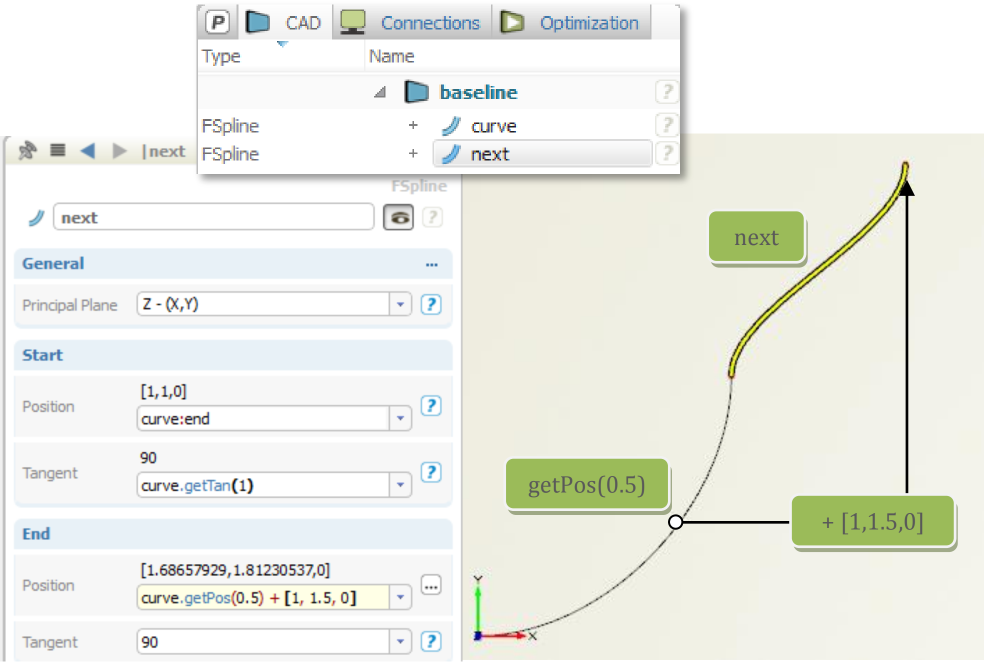
```
|curve:end
FCurvePoint
x: 1
y: 1
z: 0
```

3 Curve Positions, Lengths, etc.

All curves in CAESES provide basic functions that can be called and utilized in the modeling process. These functions can be typed into the console widget of CAESES or directly used to relate objects to each other. In commands, the principal axes x, y and z are represented by index 0, 1, and 2, respectively:

▶ curve.getPos(0.5)	[0.68,0.31,0]	xyz-coordinates at t=0.5
▶ curve.getPos(0.5).getY()	0.31	y-coordinate at t=0.5
▶ curve.getLength()	1.55	total curve length
▶ curve. getLengthFromTo(0, 0.5)	0.77	length from t=0 to t=0.5
▶ curve.getMax(1)	1	maximum y-value (0=x, 1=y, 2=z)

See the following screenshots: Another *f-spline* curve called “next” is created and joined to our first curve. The new curve uses the information of the first curve (end position & tangent angle) to set up the start position. The end position of the new curve is also based on the first curve where a vector is added to an inner position of the first curve.



Type	Name
	baseline
FSpline	curve
FSpline	next

FSpline

next

General

Principal Plane: Z - (X,Y)

Start

Position: [1,1,0] curve:end

Tangent: 90 curve.getTan(1)

End

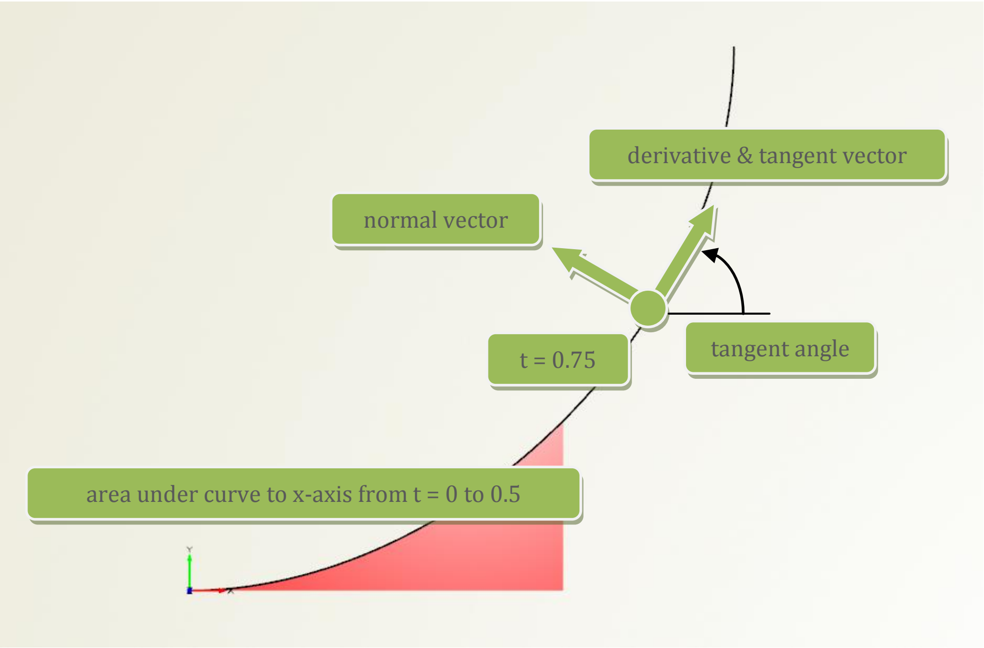
Position: [1.68657929,1.81230537,0] curve.getPos(0.5) + [1, 1.5, 0]

Tangent: 90

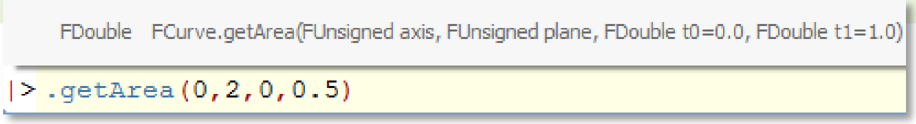
4 Angles and Derivatives

Again, the principal axes x, y and z are represented by index 0, 1, and 2, respectively. Principal planes are given in the same manner, e.g. the xy-plane has the index 2 because of the z-normal (z = index 2). Here are some more command examples for calculating curve angles and derivatives:

▶	curve.getTan(0.75, 2)	64.9	tangent angle at t = 0.75 wrt xy-plane
▶	curve.getTanVec(0.3)	[0.87,0.49,0]	normalized tangent vector at t = 0.3
▶	curve.getNormal(0.75)	[-0.91,0.42,0]	normal vector at t = 0.75
▶	curve.getDeriv(0.75, 1)	[0.66,1.41,0]	first derivative at t = 0.75
▶	curve.getArea(0, 2, 0, 0.5)	0.07	integral area wrt x-axis (0) in xy-plane (2) from t = 0 to t = 0.5


derivative & tangent vector
normal vector
t = 0.75
tangent angle
area under curve to x-axis from t = 0 to 0.5
Y
X

✓ Use auto-completion CTRL+SPACE while typing commands. This shows the command documentation, i.e. what type wil be returned (e.g. *FDouble* in the screenshot below, left) and what is expected as comannd input (e.g. *FUnsigned* axis, *FUnsigned* plane ,…). Defaulted values such as "*FDouble* t0=0.0" are automatically taken if no other values are specified. Auto-completion works for all editors, the console as well as in feature definitions (see the corresponding tutorials). The following screenshot shows how it looks in the console for a selected curve (note the first dot):



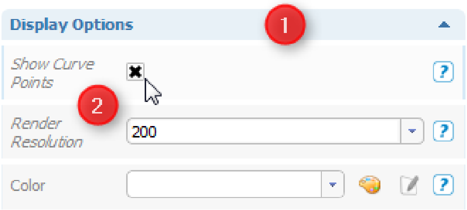
```
FDouble   FCurve.getArea(FUnsigned axis, FUnsigned plane, FDouble t0=0.0, FDouble t1=1.0)
|> .getArea(0,2,0,0.5)
```

5

Parameterization

The *parameterization* of the curve determines the “speed” of the parameter “t”. For instance, for leading edges of airfoils it is often required to have a dense region. This can be controlled by the parameterization.

- ▶ Switch on the curve’s rendering positions by going to category *display options* and activating *show curve points* (first, click at the category header to show secondary options).
- ▶ In the *general* category of a curve, change the *parameterization* and take a look at the rendering positions again.

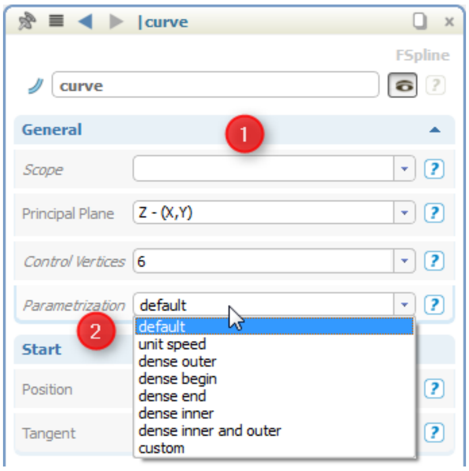


Display Options

Show Curve Points ☒

Render Resolution 200

Color



curve

FSpline

curve

General

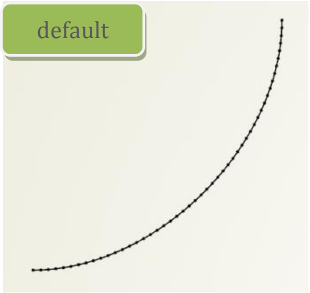
Scope

Principal Plane Z - (X,Y)

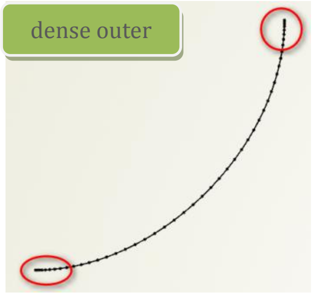
Control Vertices 6

Parametrization default

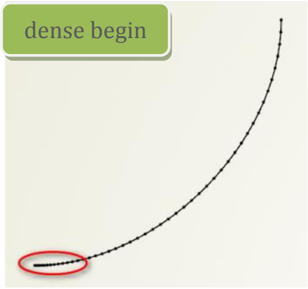
✓ For curves with the parameterization “unit speed”, the command *getPos(0.5)* will return the position at half length of the curve due to the uniform parameterization. For all other parameterizations, t = 0.5 does not necessarily correspond to the “center” position.



default

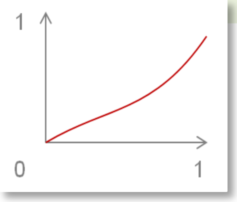


dense outer



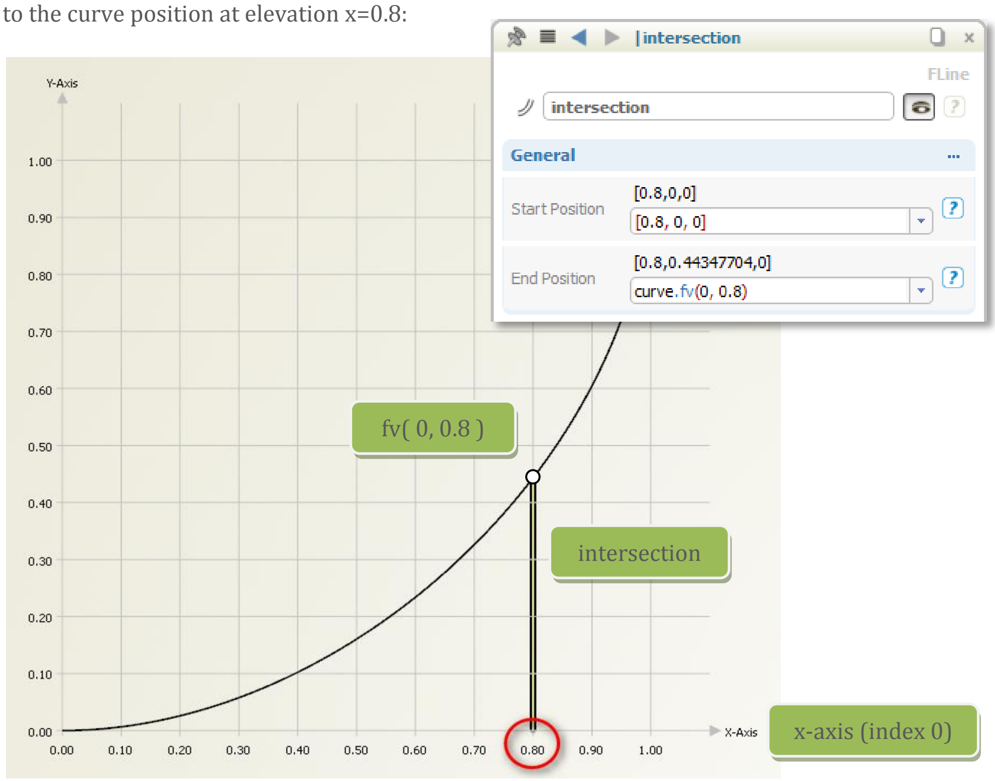
dense begin

✓ The different parameterizations allow creating dense parameterizations at the beginning or end of a curve etc. Advanced tip: Customized functions that define an individual curve parameterization can be provided by the user (parameterization option “custom”). The strictly monotonic function (any curve) needs to be given in the xy-plane in the range x=[0,1] and y=[0,1].



6 Intersection Commands

In some situations it is helpful to receive the ordinate value of a curve for a given abscissa value. Alternatively, this can be considered as an intersection. Here is an example where a line is created (*CAD > curves > line*) and gets connected to the curve position at elevation x=0.8:



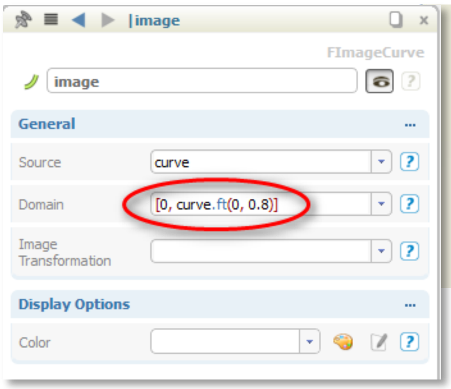
intersection

Start Position [0.8,0,0]
[0.8, 0, 0]

End Position [0.8,0.44347704,0]
curve.fv(0, 0.8)

The command “fv(axis, elevation)“ expects an axis index (0 for x) plus the elevation along this axis (here: “0.8”). It returns the vector position at the curve which can be used for modeling.

In the same way, the command “ft(axis, elevation)” returns the curve parameter “t” of the intersection. This can be used, for instance, to create sub curves using *image curves* (*CAD > curves > image curve*).



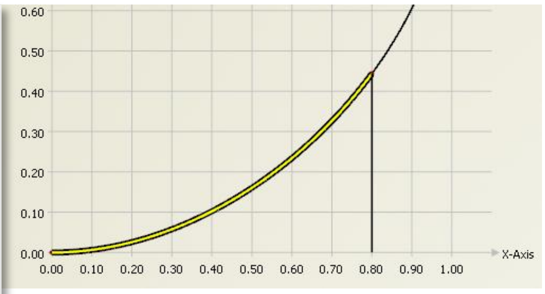
image

Source curve

Domain [0, curve.ft(0, 0.8)]

Image Transformation

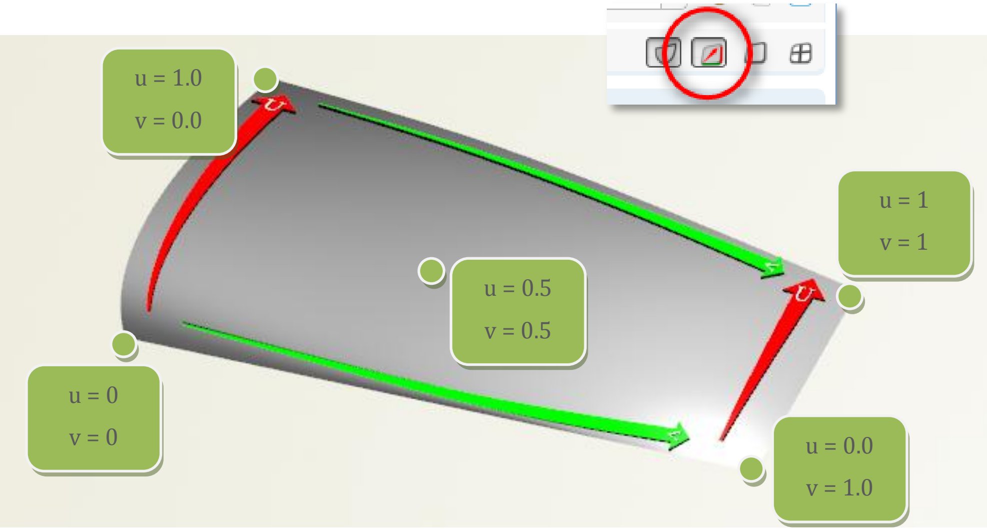
Color



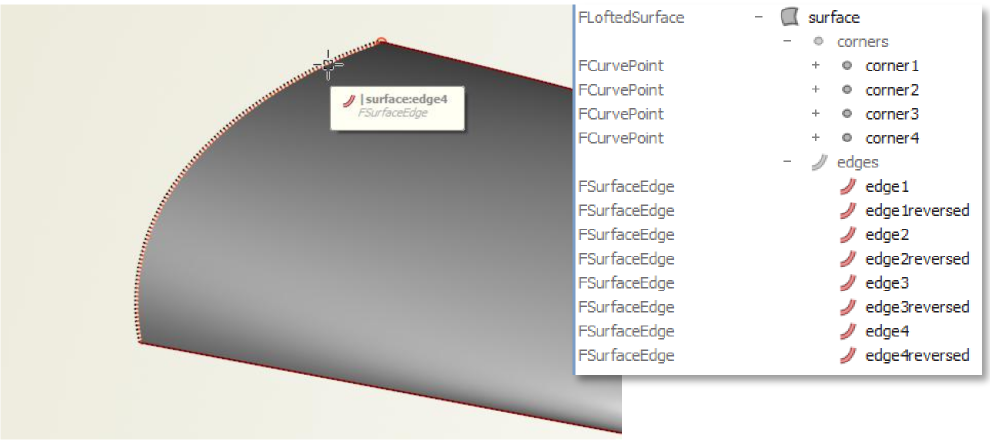
7 Surfaces

Basically, surfaces are similar to curves when considering their commands. They are also parameterized, but instead of addressing a parameter “t”, the surface parameters are called “u” and “v”. Both run in the interval [0,1].

- You can visualize the uv-parametrization in the *display options* of surfaces (click on the header of *display options* first to show secondary options).
- Example: The command “surface.getPos(0.5, 0.5)” returns the xyz-coordinates at u=0.5 and v=0.5 as a vector.



- *Surface edges* and the *corner points* are also directly accessible for all surface types either in the *CAD* tree (right below the surface object node) or in the 3D view (move the mouse over the surface edges and corner positions).

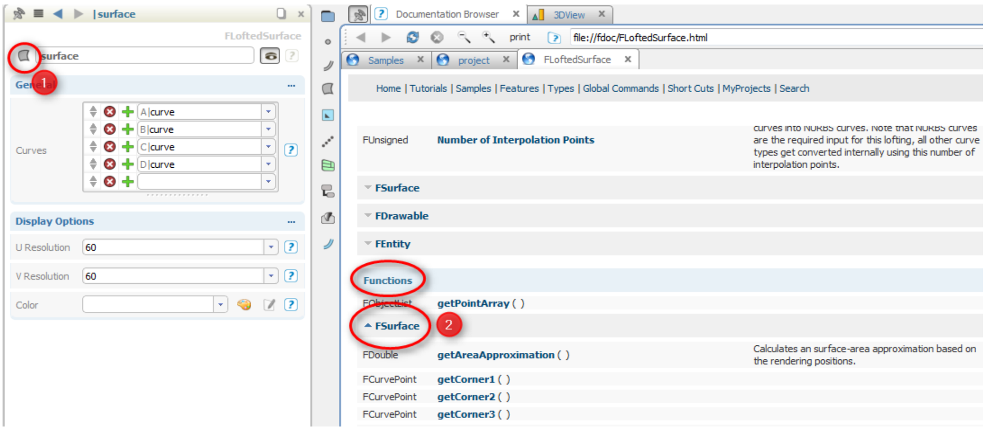


```
FLoftedSurface    –  surface
                     –  corners
FCurvePoint             +  corner1
FCurvePoint             +  corner2
FCurvePoint             +  corner3
FCurvePoint             +  corner4
                     –  edges
FSurfaceEdge               edge1
FSurfaceEdge               edge1reversed
FSurfaceEdge               edge2
FSurfaceEdge               edge2reversed
FSurfaceEdge               edge3
FSurfaceEdge               edge3reversed
FSurfaceEdge               edge4
FSurfaceEdge               edge4reversed
```

8 Additional Information

See the *type documentation* of *FCurve* and *FSurface* to get an overview of the available commands.

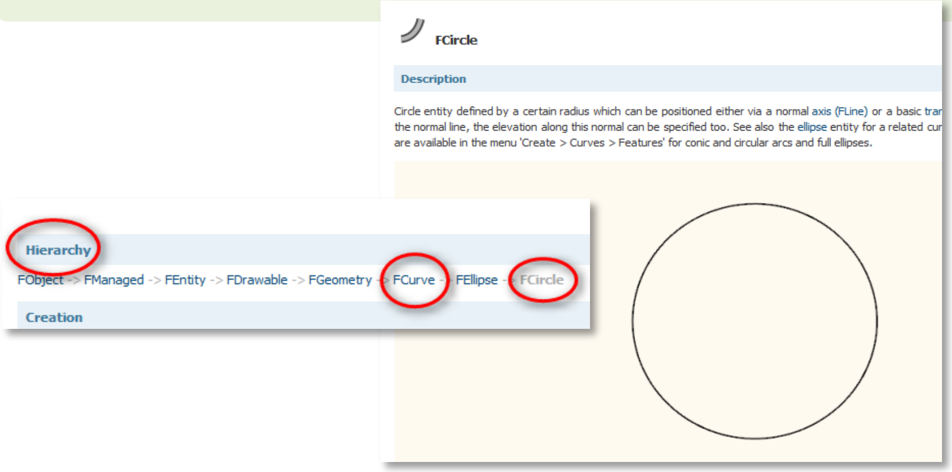
- Click on the curve or surface icon in the upper left corner (object editor) of a selected curve or surface object to show the corresponding type documentation.
- Click on *FSurface* (or for curves: *FCurve*) in the section “functions”.



FUnsigned	**Number of Interpolation Points**	curves into NURBS curves. Note that NURBS curves are the required input for this lofting, all other curve types get converted internally using this number of interpolation points.
FObjectList	**getPointArray** ( )	
FSurface		
FDouble	**getAreaApproximation** ( )	Calculates an surface-area approximation based on the rendering positions.
FCurvePoint	**getCorner1** ( )	
FCurvePoint	**getCorner2** ( )	
FCurvePoint	**getCorner3** ( )	

See also the samples in the documentation browser. Many of them utilize curve and surface functions (commands) in the geometry model.

✓ Remember that there is a type hierarchy: For instance, a circle (type *FCircle*, menu *CAD > curves > circle*) provides functions for setting and getting the circle radius, e.g. *getRadius()*. In addition, the circle also has all general curve functions received from type *FCurve* such as *getPos()* and *getTan()*. This is because *FCircle* is also a general curve, based on the type *FCurve*.



Circle entity defined by a certain radius which can be positioned either via a normal axis (FLine) or a basic tran
the normal line, the elevation along this normal can be specified too. See also the ellipse entity for a related cur
are available in the menu 'Create > Curves > Features' for conic and circular arcs and full ellipses.

FObject -> FManaged -> FEntity -> FDrawable -> FGeometry -> FCurve -> FEllipse -> FCircle