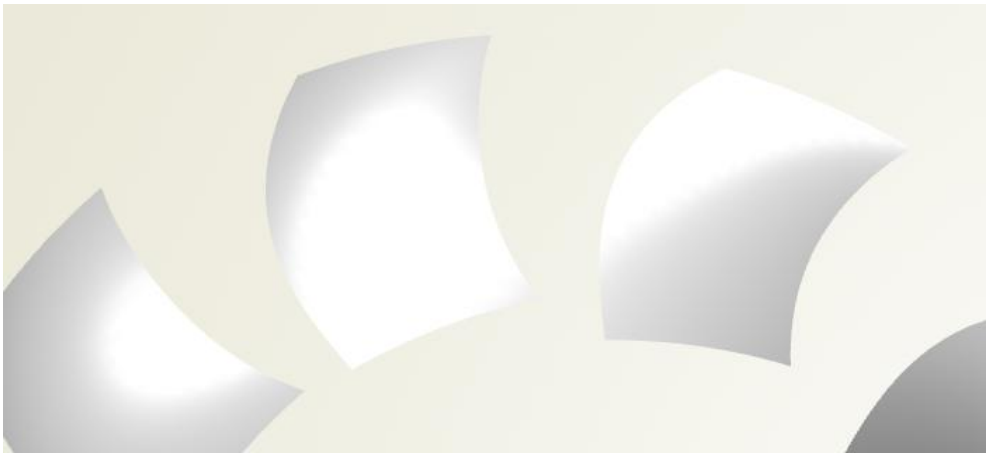
Cylindrical Transformation

The *generic blade* from the previous tutorial maps 2D profiles onto a cylindrical surface. This specific transformation is also separately available in CAESES and can be used in combination with *meta surfaces*. It represents a more generalized way for setting up "blade-like" surfaces. For instance, some designers have their own nomenclature for radial functions (the *generic blade* uses rake, skew and pitch). Additionally, the cylindrical transformation can be used to add basic support geometries to your blade which can then be used for example to define the computational domain for CFD or meshing processes.



The focus of this tutorial is not profile design but understanding the process of setting up a blade with the *cylinder transformation*. Therefore, we will create a very simple profile that represents any kind of (complex) profile definition.

CAESES Project

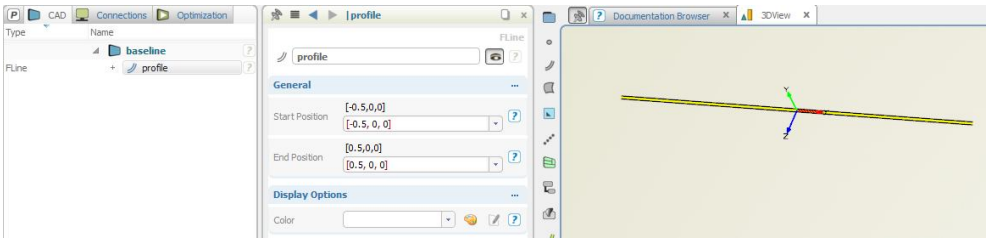
The resulting model can be found in the section *samples > tutorials* of the documentation browser.

1 Profile

In order to teach the concept of using the cylinder transformation entity, we will quickly create a representative profile.

✓ Important: The 2D definition of the profile needs to be given in the global xy-plane. The values along the x-axis will then be interpreted as rθ-coordinates, while the values along the y-axis will be interpreted as the z-coordinates of the *cylinder transformation* (see step 2).

- Create a line via *CAD > curves > line* and name it "profile".
- Set the *start position* to "[-0.5,0,0]".
- Set the *end position* to "[0.5,0,0]".

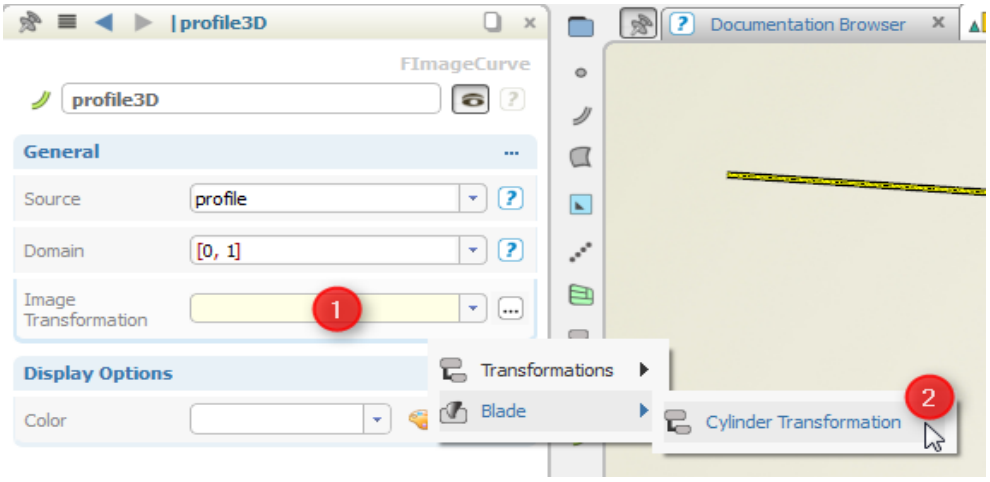


✓ We can also use 3D points (*F3DPoint*) instead of simple vector data for the positions. When it comes to *meta surfaces*, vectors can accelerate the curve generation process within features: Vectors hold only the values while 3D points are "managed" objects with a set of attributes such as name, scope, color). In our case this is not relevant but for feature definitions with many points, this effect sums up and slows down the curve generation process. Therefore, use simple vectors if a point is not necessarily required. It also keeps your project slim.

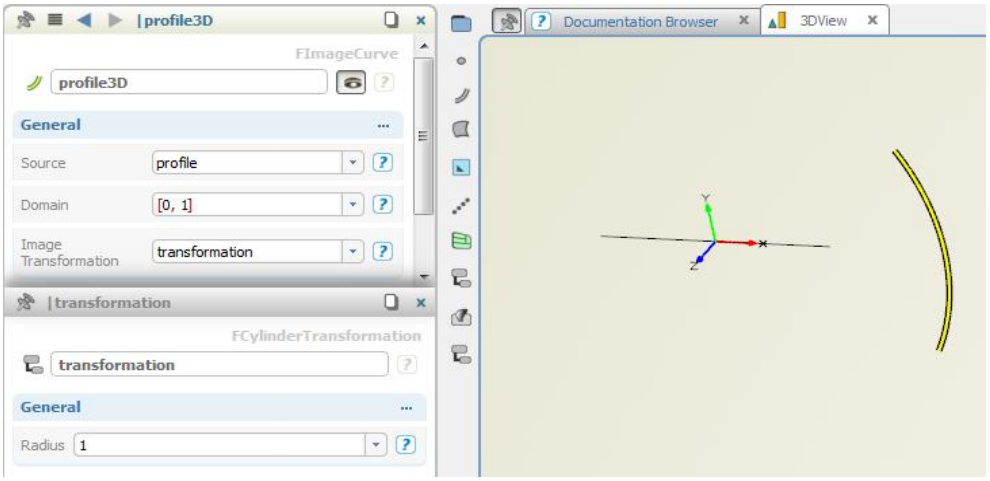
2 Image Curve and Transformation

The initial profile will be mapped into 3D space using the cylinder transformation:

- Make sure that the initial curve “profile” is selected.
- Create an image curve via *CAD > curves > image curve* and name it “profile3D”.
- Create a *cylinder transformation* by choosing “create …” from the pull-down menu of the attribute *image transformation* (alternatively, via *CAD > blade > cylinder transformation*).



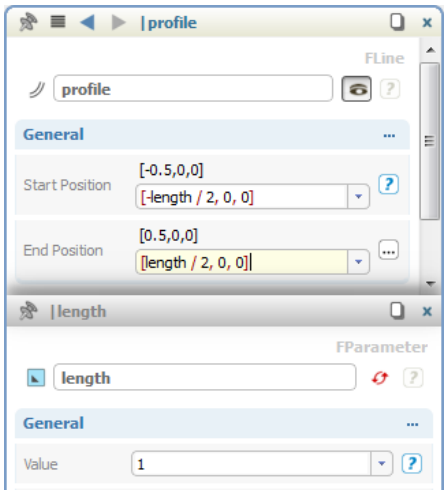
- Set the name of the new transformation object to “transformation.”



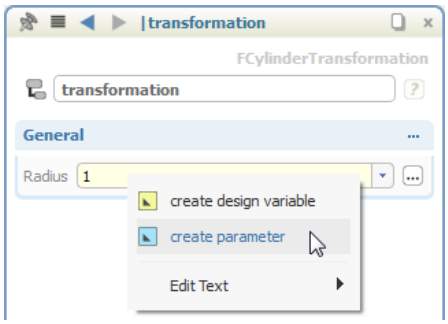
3 Parameters

You can now change the radius of "transformation" and directly see how your profile is mapped into 3D space. In this step, a length and a radius parameter are introduced since these will be the variable values for our upcoming feature definition.

- Mark the value "0.5" of the profile's start position and create a *parameter* via the context menu (right mouse button).
- Set the name of the new parameter to "length".
- Set the parameter also for the x-coordinate of the end position.
- Divide the expressions by "2" so that the parameter controls the length of the total curve (ensure "-" precedes "length" at the start position).
- Reset the "length" value to "1".



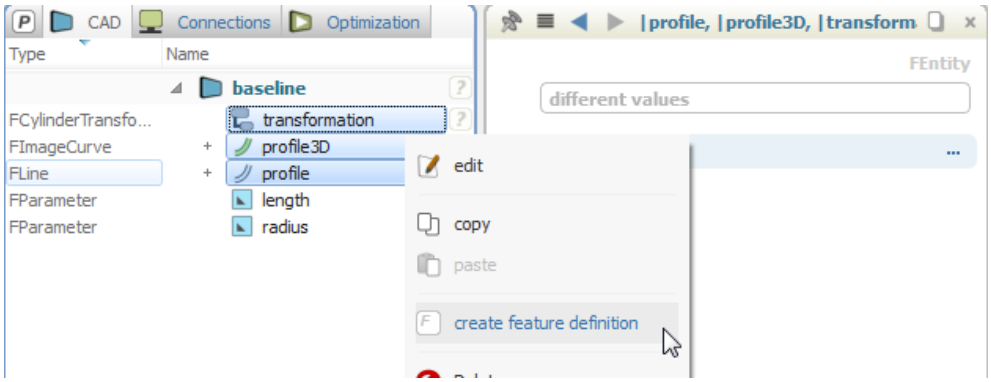
- Create a parameter for the radius of "transformation" via the context menu.
- Set the name of the new parameter to "radius".



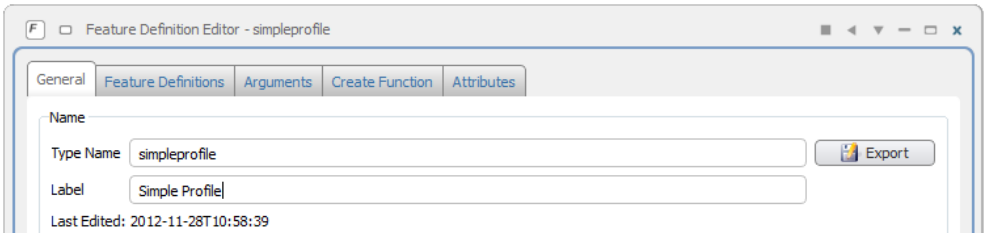
4 Feature Definition

The curve is variable according to its length and the radius value. Since we will use a *meta surface* (and a *curve engine*) it requires a curve from a feature definition (also see the meta surface tutorials). Let's now create the feature definition:

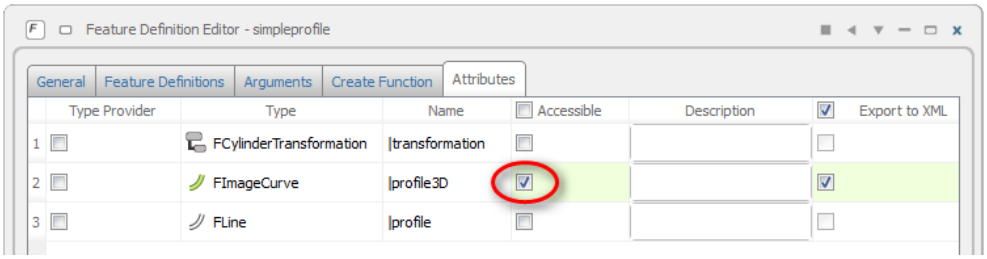
- Select the curves and the transformation and create a *feature definition* via the context menu of the selection:



- Set the *type name* to "simpleprofile" (*general* tab).



- In the *attributes* tab, make only "profile3D" accessible – this is our curve of interest for the upcoming *meta surface*.



- Press *apply* and close the dialog.

5 Meta Surface

In this step, a *meta surface* is created. A simple, say, “metal sheet” results from the basic shape of the profile:

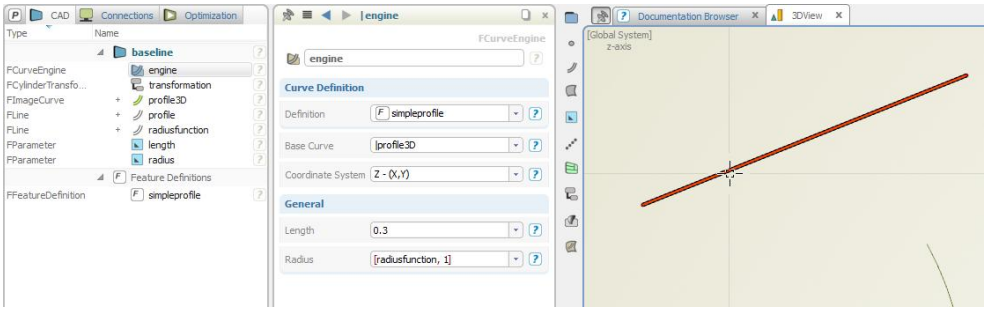
- Select the definition “simpleprofile” (in the *CAD* tree, node *feature definitions*).
- Create a *curve engine* via *CAD > curves > curve engine* and set its name to “engine”.

✓ The creation of the curve engine involves the information of the selection and automatically sets the *definition* as well as the *base curve*.

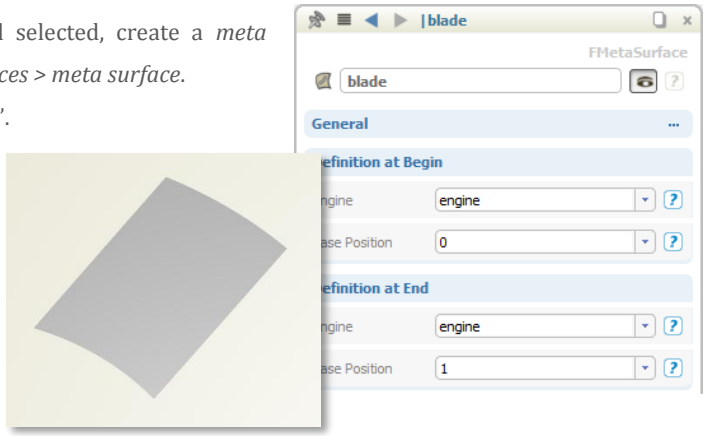
- Set the attribute length to a constant value of “0.3”.

Let’s say the hub radius is “0.6” and the tip radius “1”. The corresponding radial function can be created and applied as follows:

- Create a line via *CAD > curves > line* and name it “radiusfunction”.
- Set the start position to “[0,0.6,0]”.
- Set the end position to “[1,1,0]”.
- Select “engine” and set this function at the attribute *radius*.



- While “engine” is still selected, create a *meta surface* via *CAD > surfaces > meta surface*.
- Set the name to “blade”.

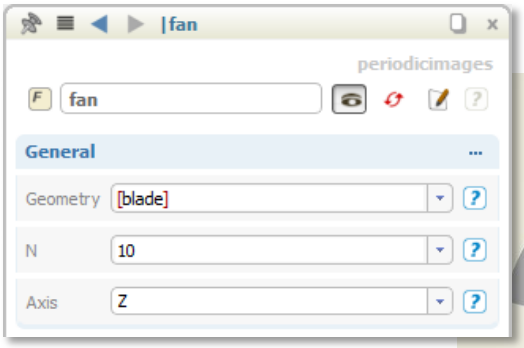


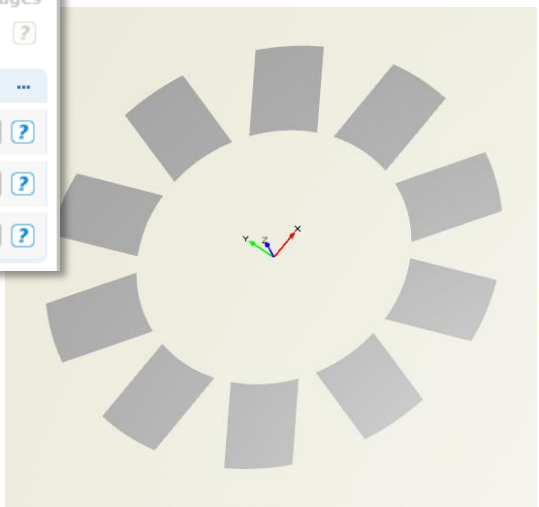
6 Periodic Visualization

For visualization purposes, we create “copies” of the blade by using a predefined feature definition:

- Select the surface “blade”.
- Choose *CAD > transformations > more > periodic images*.
- Set the number of blades (*N*) to “10”.
- Set axis to “Z”.
- Press the *create* button.
- Rename the new object “f1” to “fan”.

✓ Remember: This is only for visualization. The feature does not create new surfaces but only so-called *GL-images* (type FGlImage) which cannot be exported. If you need all blades to conduct a CFD analysis then you need to create *image surfaces* (type FImageSurface).



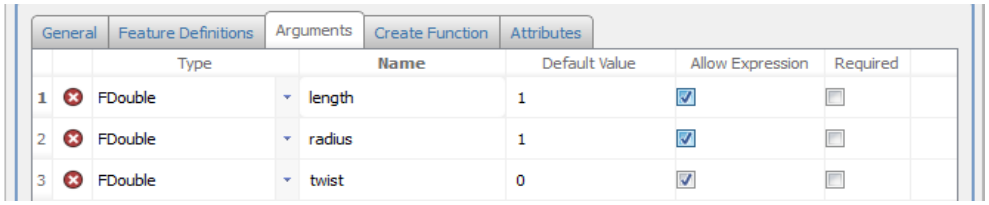


7

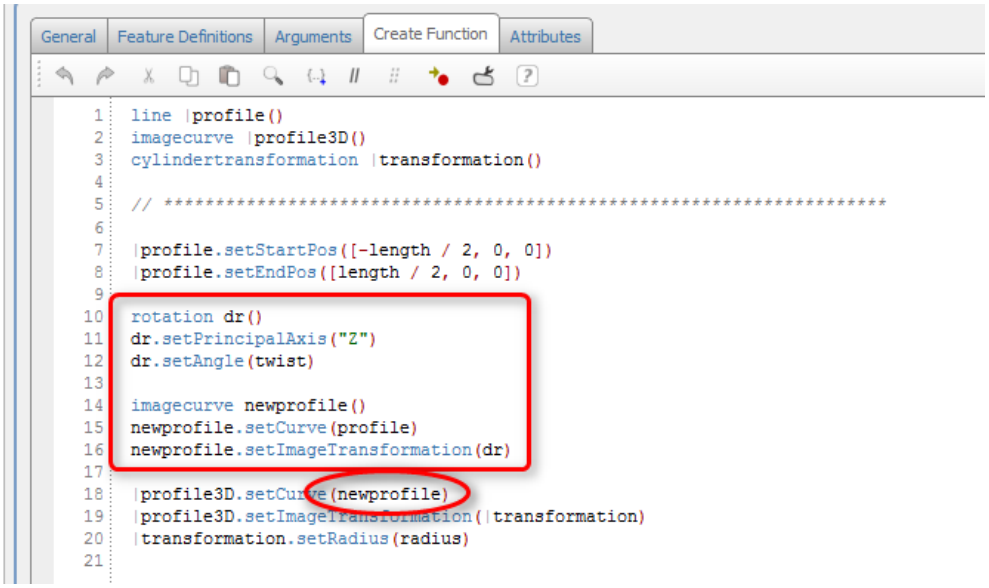
Adding Blade Parameters

So far, our fan is not that exciting since only the profile length can be controlled. In this step, a twist parameter for the blade is introduced. Basically, any new blade parameter needs to be defined in the 2D space (xy-system) of the profile. In order to twist the blade, a rotation needs to be applied to the initial profile.

- Edit the feature definition "simpleprofile" (e.g. by double-clicking on it).
- Add another input argument of type *FDouble* and name it "twist".
- Set the default value to "0".



- In the *create function*, add the following red-labeled lines to your feature definition:



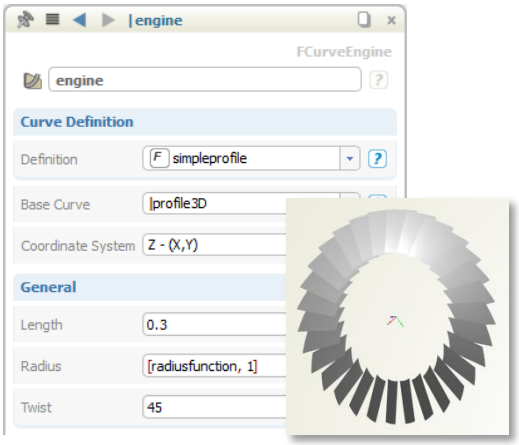
- Press *apply* and close the dialog (optional: check attribute *accessible* again, see step 4).

First, a *rotation* gets defined using "twist" as angle input. This rotation is then used in combination with the initial profile to create a new image curve. Finally, the new image curve (i.e. the rotated profile) is then mapped into the 3D space.

8 Radial Twist Function

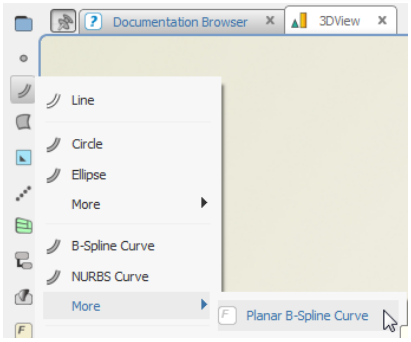
We can now change the value of the twist. Again, this is only a constant value for which we introduce a function in this step. First, a quick test:

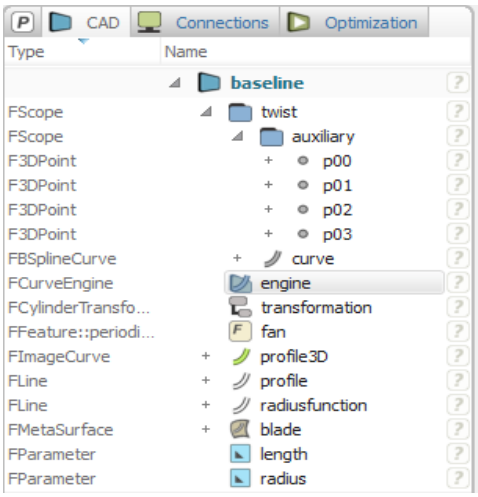
- Select “engine” and set a constant value of “45” for the new input argument *twist*.
- If you like, change the number of blades of “fan” as well for testing.

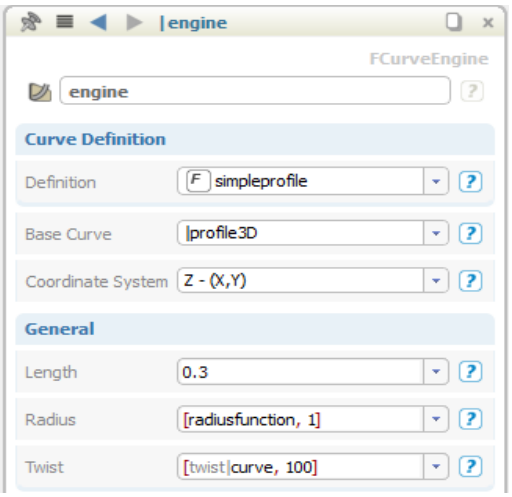


Let’s create a function for the twist.

- Similar to the previous tutorials, choose e.g. from the *b-spline* menu to create an initial planar *b-spline* curve.
- Set *number of control vertices* to “4”.
- Set *constant value* to “0.5”.
- Press *execute*.
- Rename “f1” to “twist”.
- Drag & drop the curve from scope “twist” into the curve engine attribute *twist*.
- Set a scaling factor of “100” so that the normalized values from the function are multiplied by “100” (e.g. 0.5 * 100 = 50° which is the angle for the twist rotation).



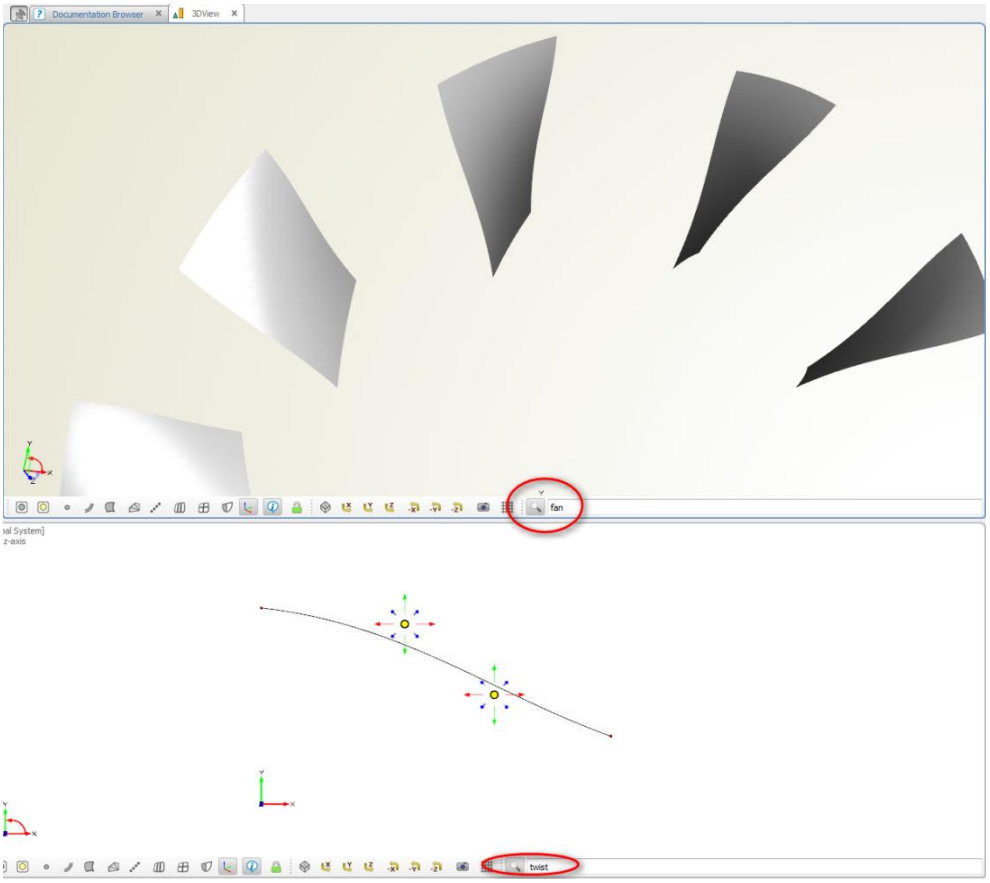




9 Twisting the Blade

The twist can now be controlled by the twist function. In order to manipulate the twist, do the following:

- Select one or more points of scope “twist|auxiliary” and drag them in x- and y-directions.
- Optionally, use two 3D views (*view > windows > 3DOverView*) and name filters of the views for distinct visualization.



10 Finalizing

Here are some final remarks for next steps and for finalizing this tutorial on your own:

- Add a radial function for the profile parameter *length* and set it at the curve engine (see step 8). A scaling factor (for *twist* it was "100") is not required at the curve engine. For instance, use copy & paste of the existing "twist" scope if you like to use the same type of curve. Or, try out different curve types such as the *fspline curve* for which the tangent angles can be controlled e.g. by *design variables*.
- Add two more input arguments for the profile definition to move it in the x- and y-directions in the 2D plane (say, "dx" and "dy"). Use a *translation* object for this purpose. The new translation and the rotation are then put into a transformation chain. This chain is then set as transformation for "newprofile":

```
7  |profile.setStartPos([-length / 2, 0, 0])
8  |profile.setEndPos([length / 2, 0, 0])
9
10 rotation dr()
11 dr.setPrincipalAxis("Z")
12 dr.setAngle(twist)
13
14 translation delta(dx,dy,0)
15
16 transformationChain total([dr,delta])
17
18 imagecurve newprofile()
19 newprofile.setCurve(profile)
20 newprofile.setImageTransformation(total)
21
22 |profile3D.setCurve(newprofile)
23 |profile3D.setImageTransformation(|transformation)
24 |transformation.setRadius(radius)
```

- Add radial functions for the new input parameters "dx" and "dy" from above.

✓ Although we specified translations in x- and y-directions (i.e. "dx" and "dy"), the actual cylindrical transformation implies that "dx" shifts in tangential direction, and "dy" shifts in the axial direction. If you like, rename "dx" and "dy" accordingly.

✓ If you want to use another profile definition (and not just a simple line), only the first part of the feature definintion needs to replaced by a new profile description. If the new profile is also called "profile", the sequence starting from line 12 in the screenshot will work correctly. Always check the *attributes* tab for changes and make sure that your final 3D curve is accessible ("profile3D").