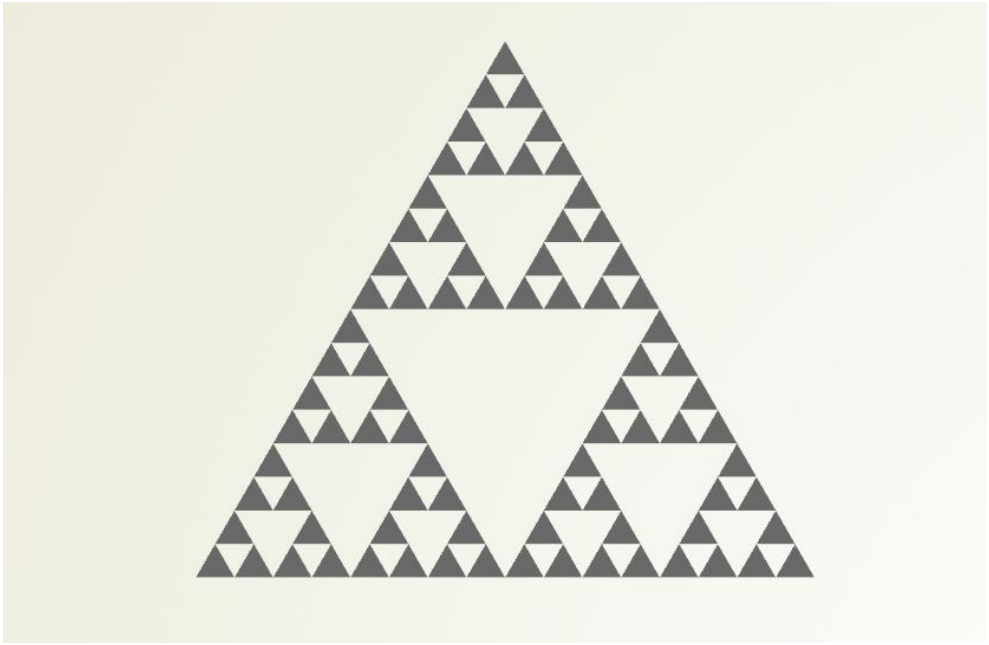
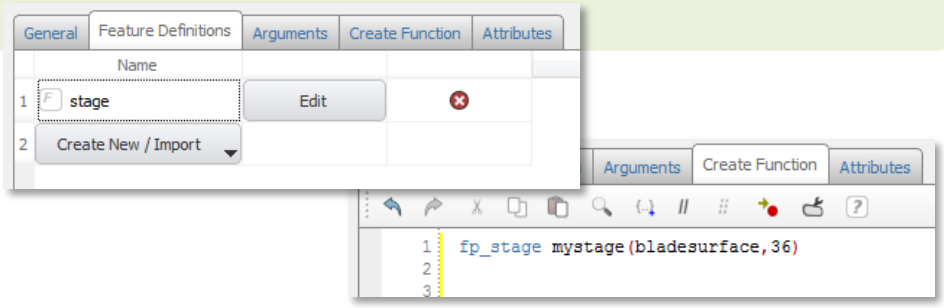
Functions

In this tutorial a feature is defined that creates the *Sierpinski* triangle. For such a mathematical structure we have to repeatedly create triangles. The subtask of creating a single triangle is defined in a function that is directly embedded in the feature definition. We can then simply call this function multiple times in a recursive sequence.

Functions can ease your work with feature definitions. In particular, they make your command sequence more readable and easier to maintain.



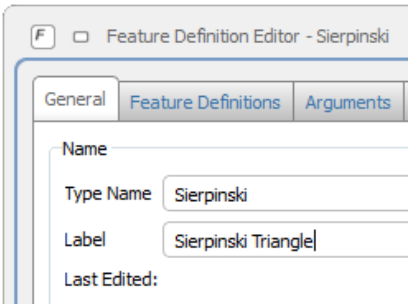
✓ See also the feature samples in the documentation browser where so-called *nested features* are demonstrated. Instead of a function, a complete feature definition can also be embedded and used in another feature definition.

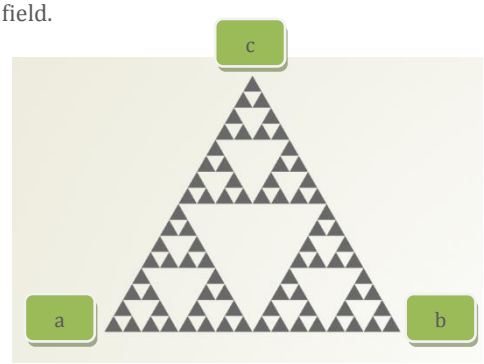


1 New Feature Definition

We start from scratch by creating a new feature definition. This definition will later create the entire Sierpinski triangle.

- Create a new definition by selecting *features > new definition.*
- Enter "Sierpinski" in the *type name* field (*general* tab).
- Enter "Sierpinski Triangle" in the *label* field.





The Sierpinski triangle needs a starting triangle (three vector positions) and a depth that defines how often the recursive triangle creation will be executed.

- Choose three *FVector3* arguments for the positions; call them *a, b* and *c* and set some default values (see screenshot).
- Choose an *FUnsigned* argument for the depth and set a default value of "3".

✓ We chose the type *FUnsigned* for "depth" since we expect only positive integer values ("unsigned integer"). If you would like to allow negative input values for your definitions, then choose the type *FInteger*.

2 Function for Triangle Creation

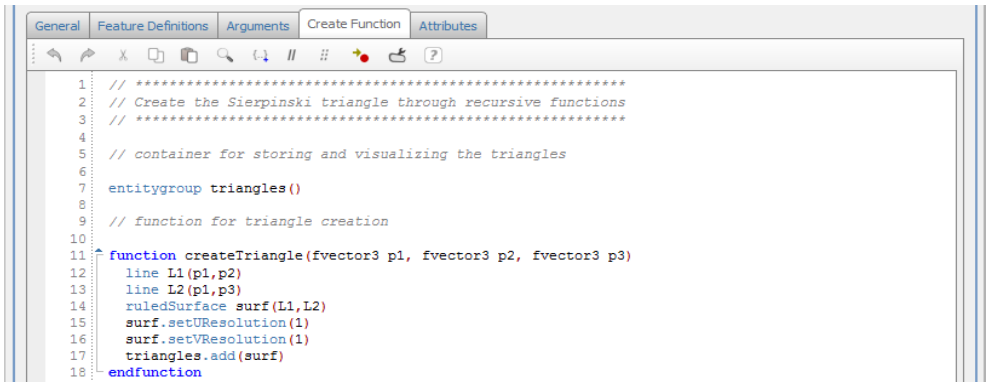
Creating a triangle is a rather simple task that can be encapsulated in a function. This is done in this step. The function takes three vector arguments “p1”, “p2” and “p3” (type *fvector3*) and creates a triangle by using a *ruled surface* between two *lines*. An entity group called “triangles” is also defined. It is our container that will store all upcoming triangles.

▶ Click on the tab *create function* and type in the following sequence:

```
entitygroup triangles()
function createTriangle(fvector3 p1, fvector3 p2, fvector3 p3)
  line L1(p1,p2)
  line L2(p1,p3)
  ruledSurface surf(L1,L2)
  surf.setUResolution(1)
  surf.setVResolution(1)
  triangles.add(surf)
endfunction
```

▶ Press *evaluate* to check that are no typing mistakes.

✓ You can copy & paste the sequence from above. Remember: If you type in commands manually, use auto-completion while typing. For instance, type “rule” + CTRL + SPACE: This provides the command for the ruled surface (next to other commands that start with “rule”).



3 Recursive Function

Next we will define the recursive function that calls the function from the previous step.

▶ Extend the definition by typing in the following sequence after the function "createTriangle()" from the previous step:

```
function splitTriangle(fvector3 p1, fvector3 p2, fvector3 p3,
unsigned currentDepth)
  if (currentDepth > depth)
    createTriangle(p1,p2,p3)
    return()
  endif
  fvector3 mid1((p1+p2)/2)
  fvector3 mid2((p2+p3)/2)
  fvector3 mid3((p1+p3)/2)
  unsigned d(currentDepth)
  d += 1
  splitTriangle(p1,mid1,mid3,d)
  splitTriangle(mid1,p2,mid2,d)
  splitTriangle(mid2,p3,mid3,d)
endfunction
```

The function "splitTriangle()" calls itself three times again as long as the depth constraint "(currentDepth > depth)" is not violated.

▶ Press *evaluate*.

✓ The command "return()" terminates the feature execution at this point.

4

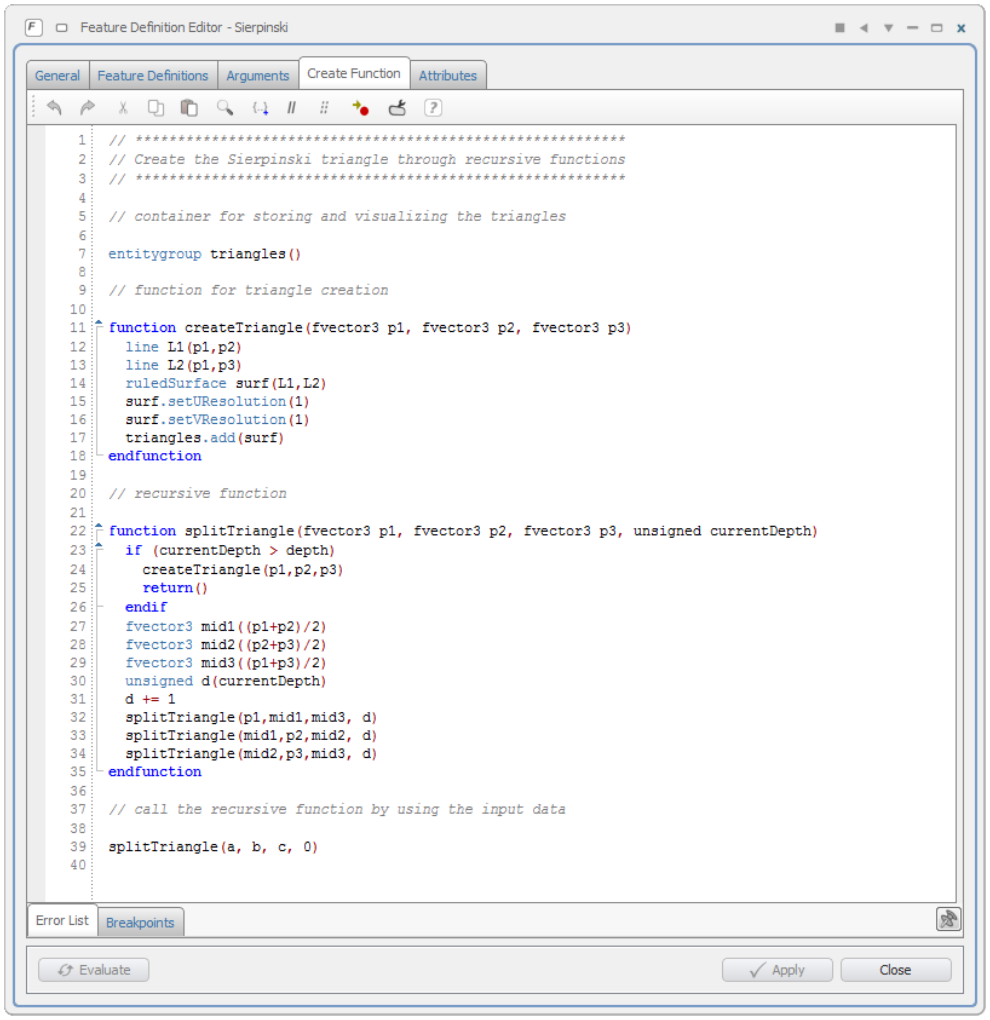
Initiate Creation of Sierpinski Triangle

So far, there are only two functions and we still have to initiate the creation of the Sierpinski triangle. Thus, we have to call the function "splitTriangle()" with the input arguments "a", "b" and "c" that are provided by the user.

▶ Type in the following after the function "splitTriangle" press *apply* and close the dialog:

```
splitTriangle(a, b, c, 0)
```

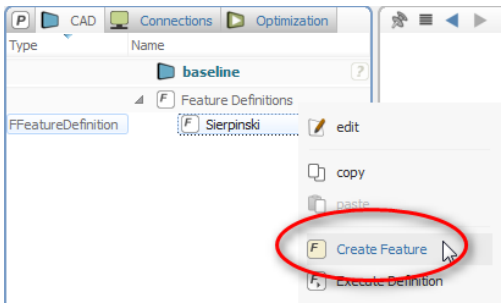
Here is a final screenshot:



5 Create a Feature

Finally, we can now create a feature from this definition in order to test our Sierpinski triangle.

- Create a feature from the context menu of the definition “Sierpinski”.



- Change the input values of the created feature “f1” in order to check whether it works correctly. Switch into the z-view (button in the 3D view).