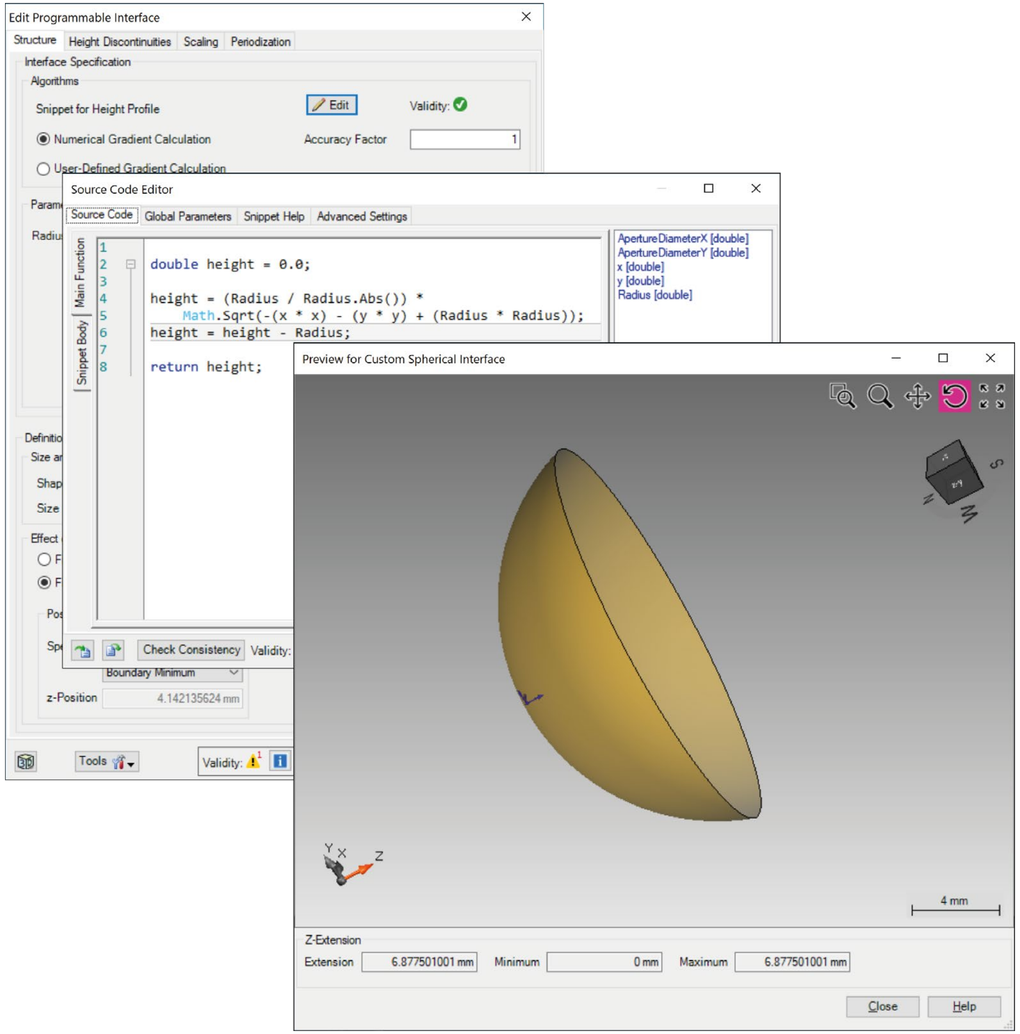
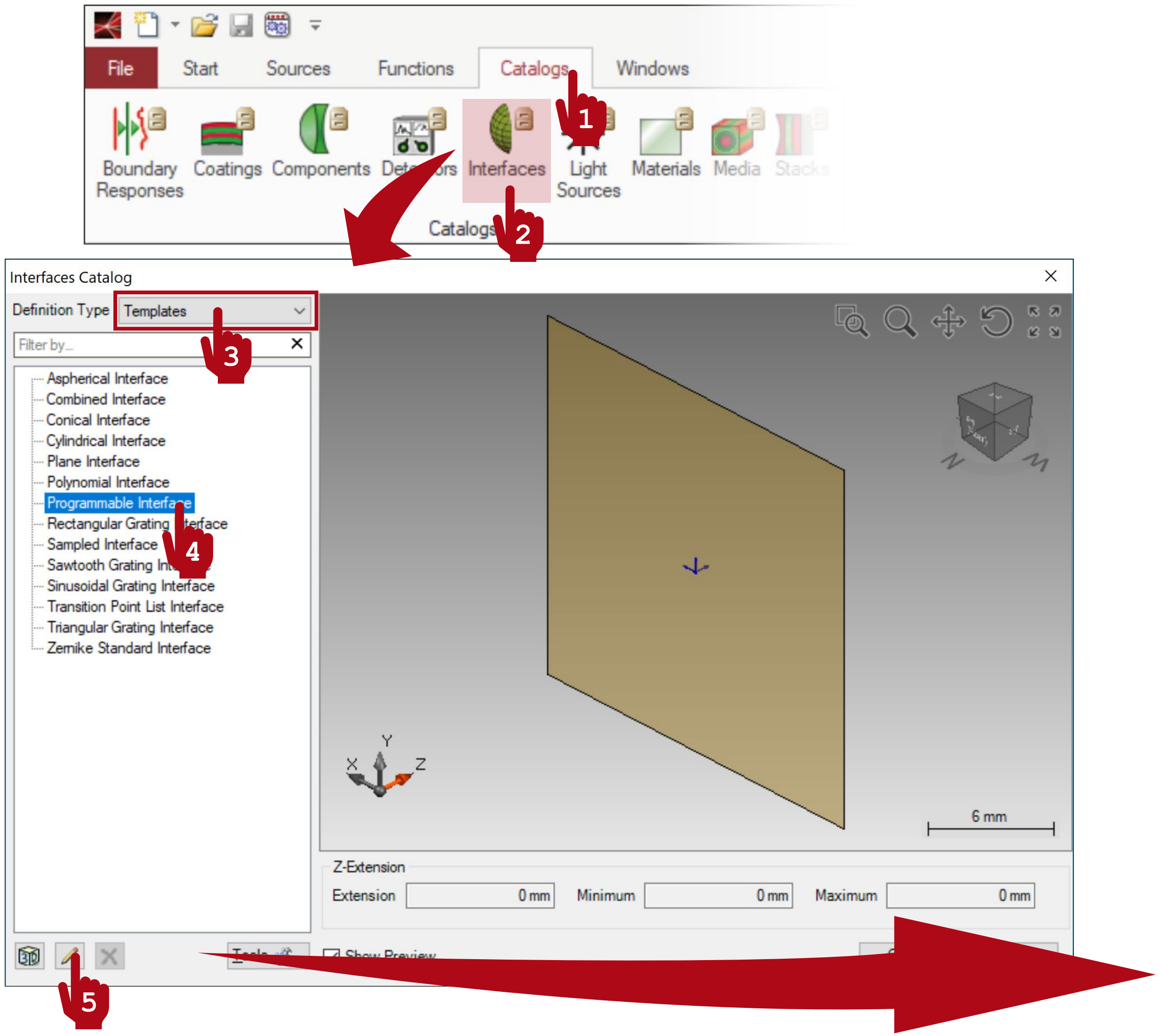
How to Work with the Programmable Interface & Example (Spherical Surface)

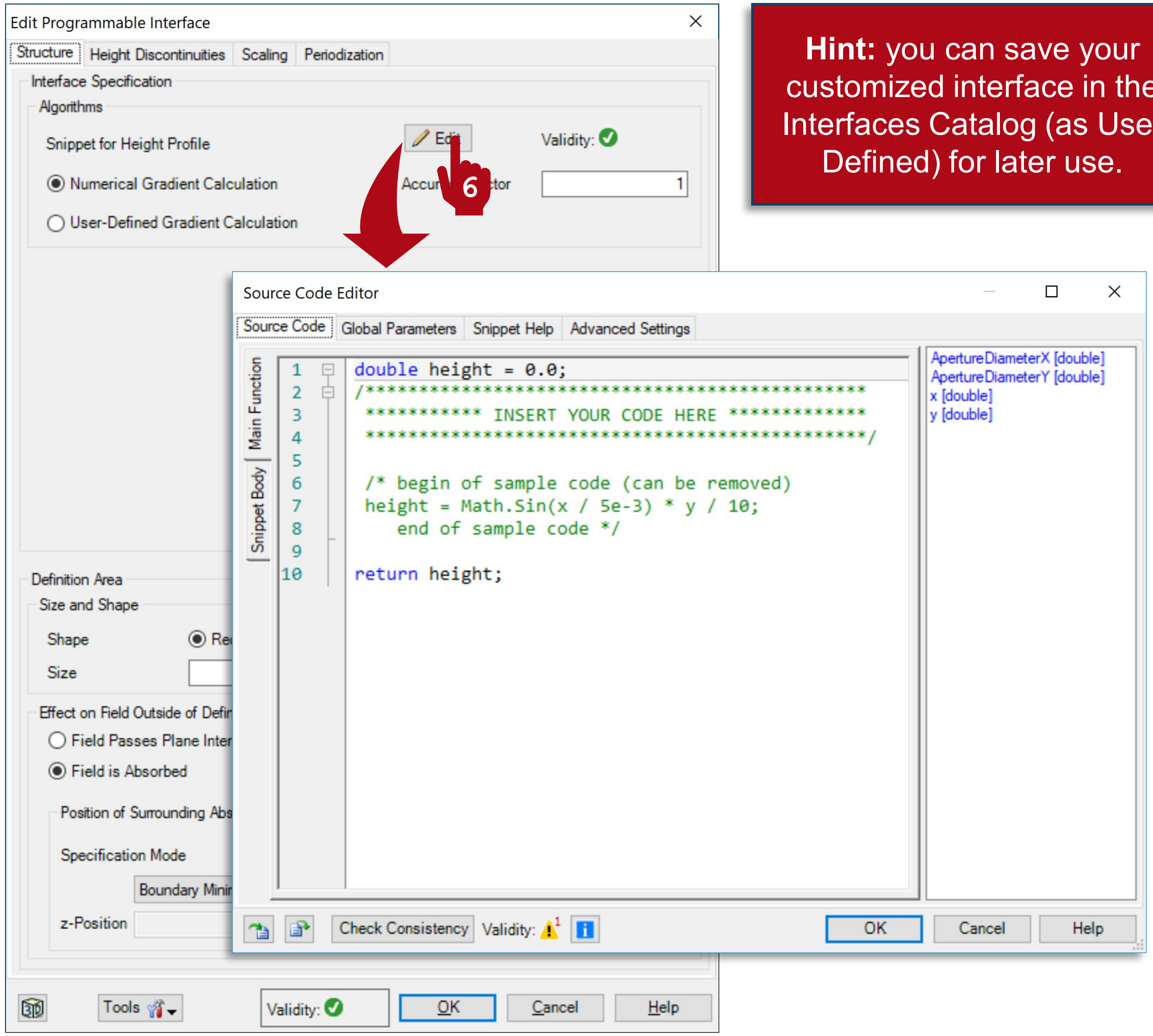
Abstract



Providing maximum versatility for your optical simulations is one of our most fundamental objectives. In this document we show you how to program custom surfaces: that is, how to define a height function *h(x, y)* that describes a 3D surface with respect to the *x, y* coordinates which span the parametrization plane. These surfaces can then be used to configure the optical components in your system. Conical surfaces, being some of the most fundamental, are of course provided as a default template in VirtualLab; in this tutorial, however, we use a spherical surface as a simple programming example.

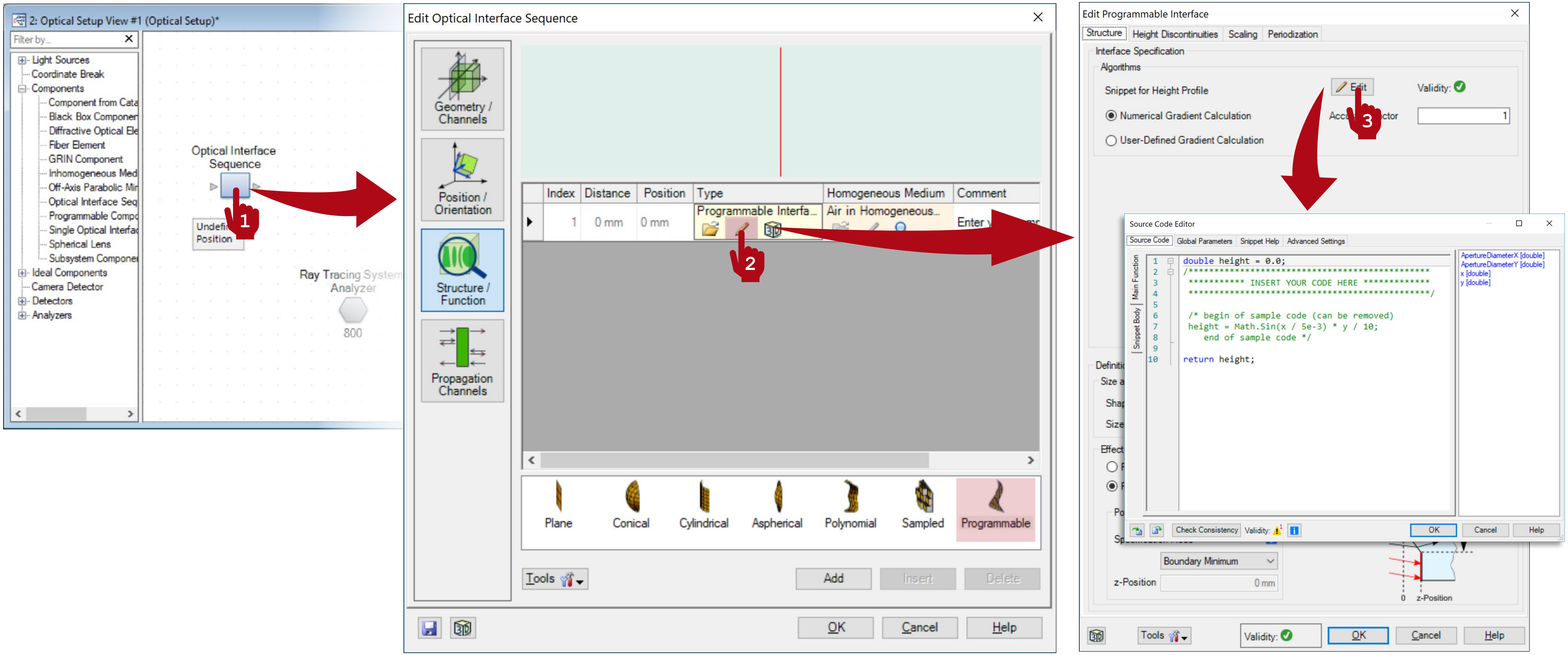
Where to Find the Programmable Interface: Catalog



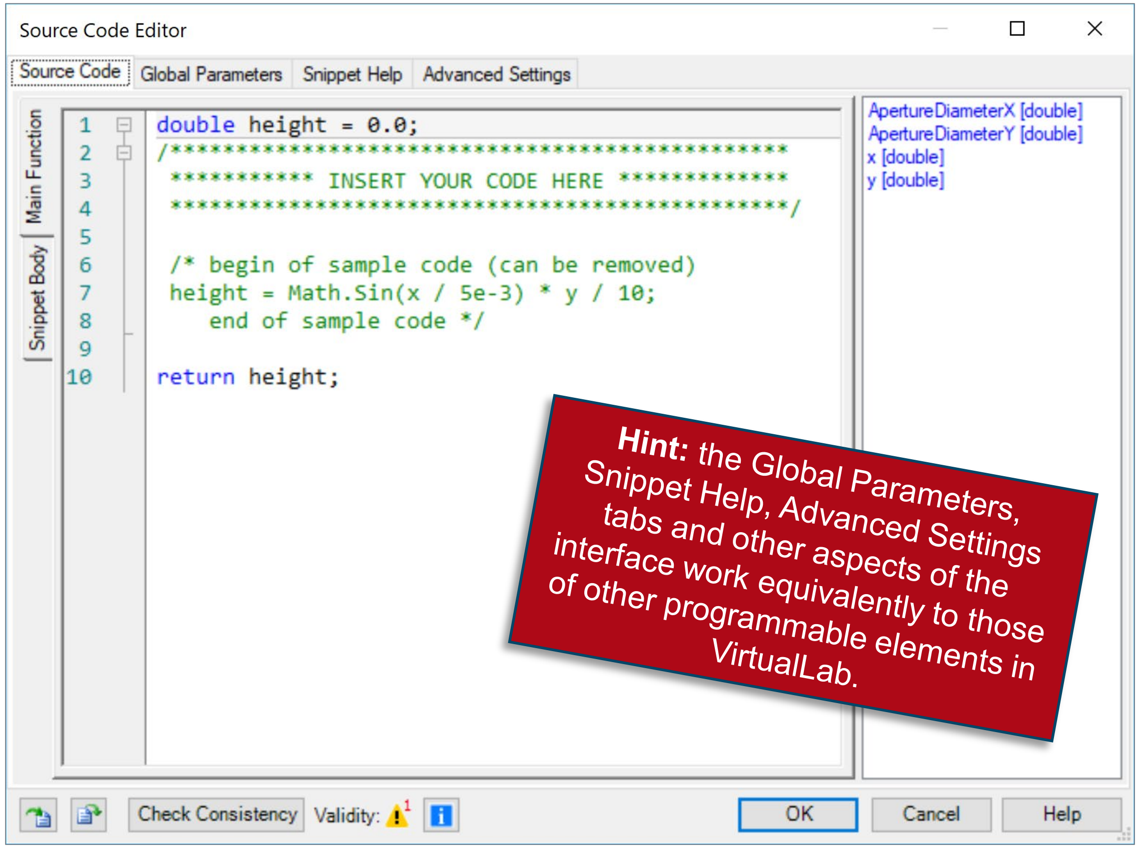


Hint: you can save your customized interface in the Interfaces Catalog (as User Defined) for later use.

Where to Find the Programmable Interface: Components



Writing the Code

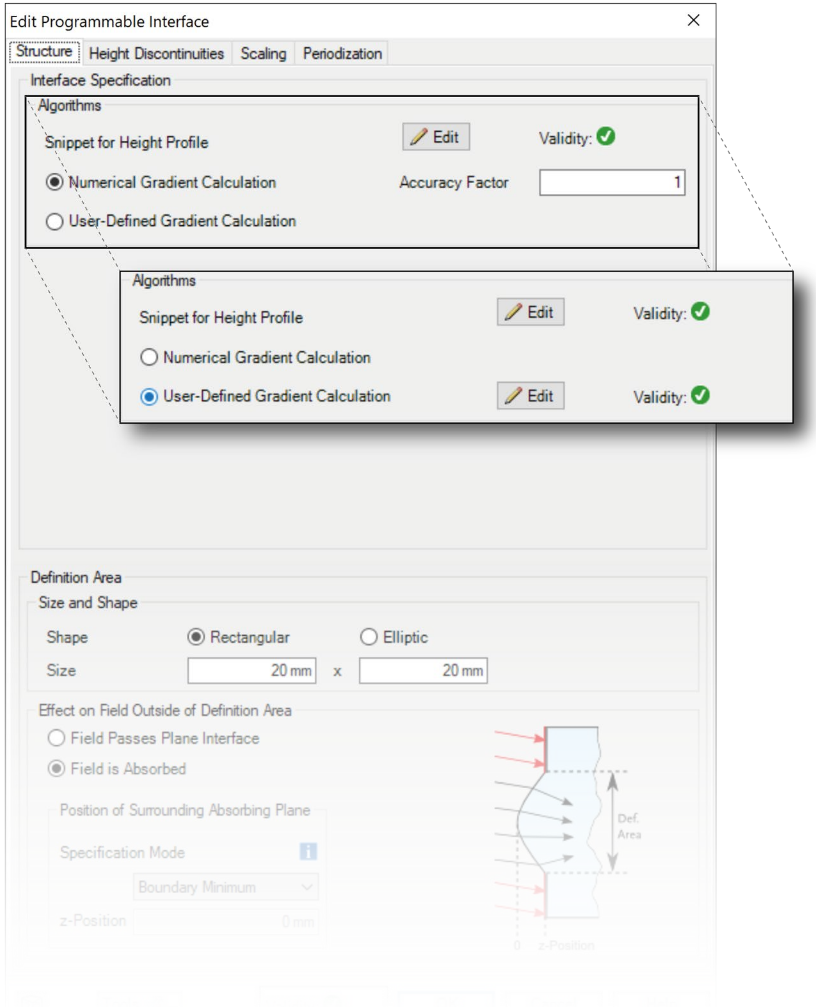


- The panel on the right shows a list of available independent parameters.
- `x` and `y` represent the independent variables, the 2D coordinates spanning a plane.
- The interval where `x` and `y` are defined is given by `ApertureDiameterX` and `ApertureDiameterY` (both of them determined in the general configuration dialog of the interface.
- The code in the Main Function must return a `double` value per `x, y` point. This value represents the height at that point. The set of all these height values defines a 3D surface.
- Use the Snippet Body to group parts of the code in support functions.

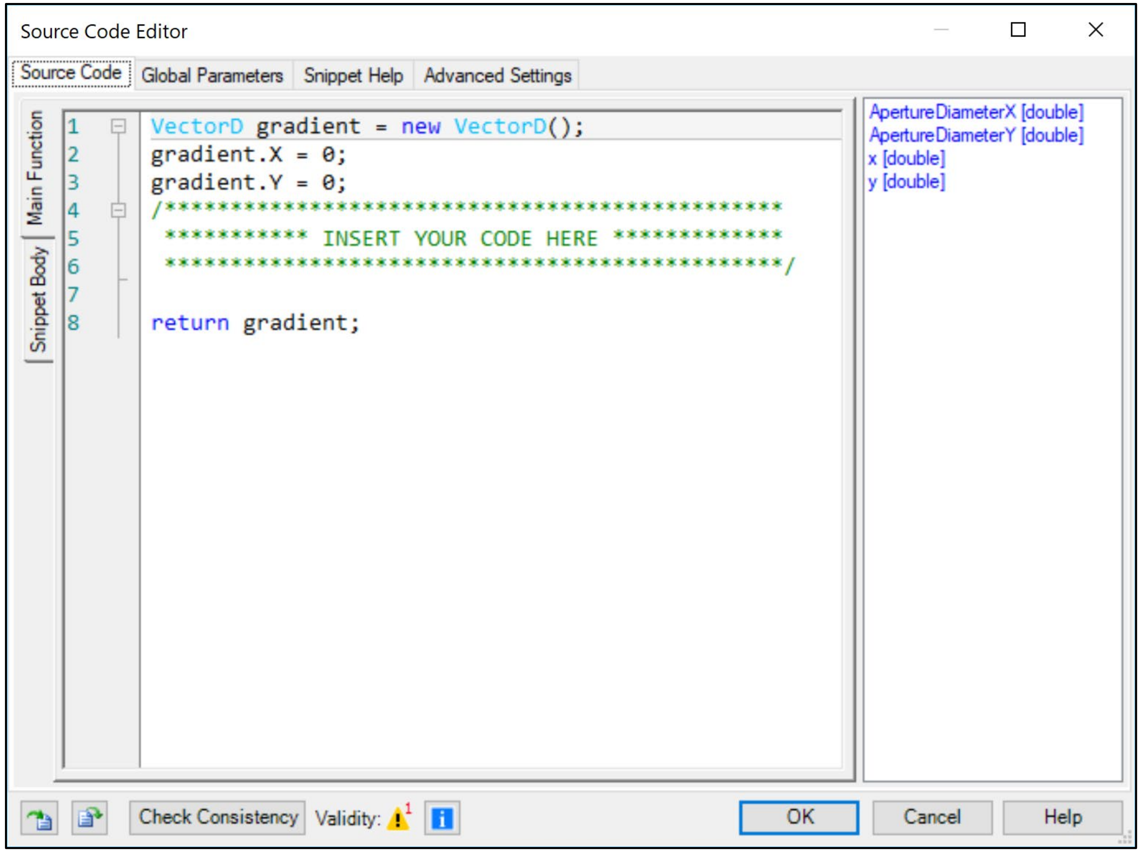
Definition of the Surface Gradient

- An accurate calculation of the gradient of the interface is fundamental in an optical simulation.

- The Programmable Interface in VirtualLab allows for two different modes of definition of the gradient: numerical, with adjustable accuracy (automatically carried out by the software) or analytical (programmed additionally by the user).



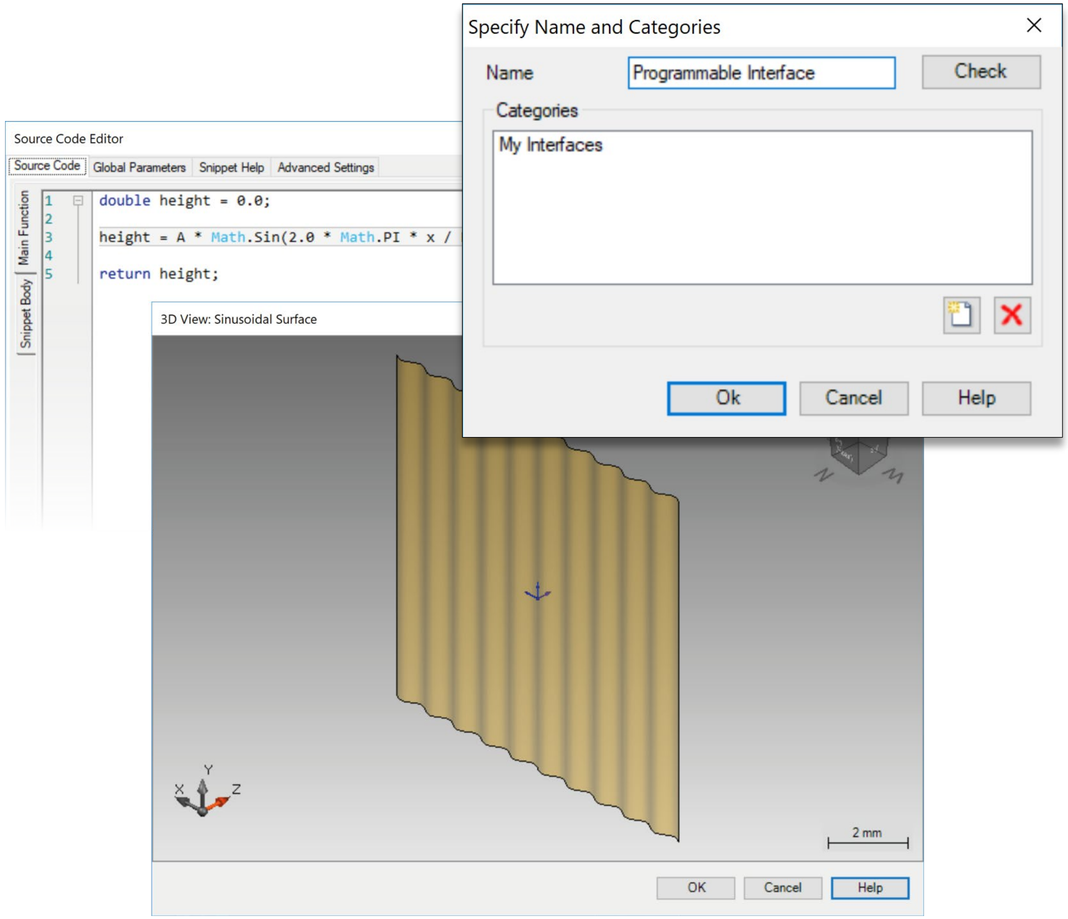
User-Defined Surface Gradient



- The panel on the right shows the same list of available independent variables.
- This time, the code in the Main Function must return a `VectorD`—a vector with two `double` (real-valued) coordinates. Use the Snippet Body to group parts of the code in support functions.
- Using the **analytical definition** of the gradient where possible makes for the **more accurate** alternative. The consistency of the code is checked by the software, but the user must make sure that the function entered for the gradient coincides with the corresponding interface!

Output of the Programmable Interface

- The output is a 3D surface which can be used to define the interface of an actual optical component in a system.
- The custom interface can be programmed directly within the real component where it is required.
- Or, alternatively, it can be saved in the catalog for later use.



Programming a Spherical Surface

The Spherical Surface

A sphere is defined as the locus of all the points *x, y, z* located at the same distance *R* from a common centre (which we shall place at the origin)

$$x^2 + y^2 + z^2 = R^2 \,. \tag{1}$$

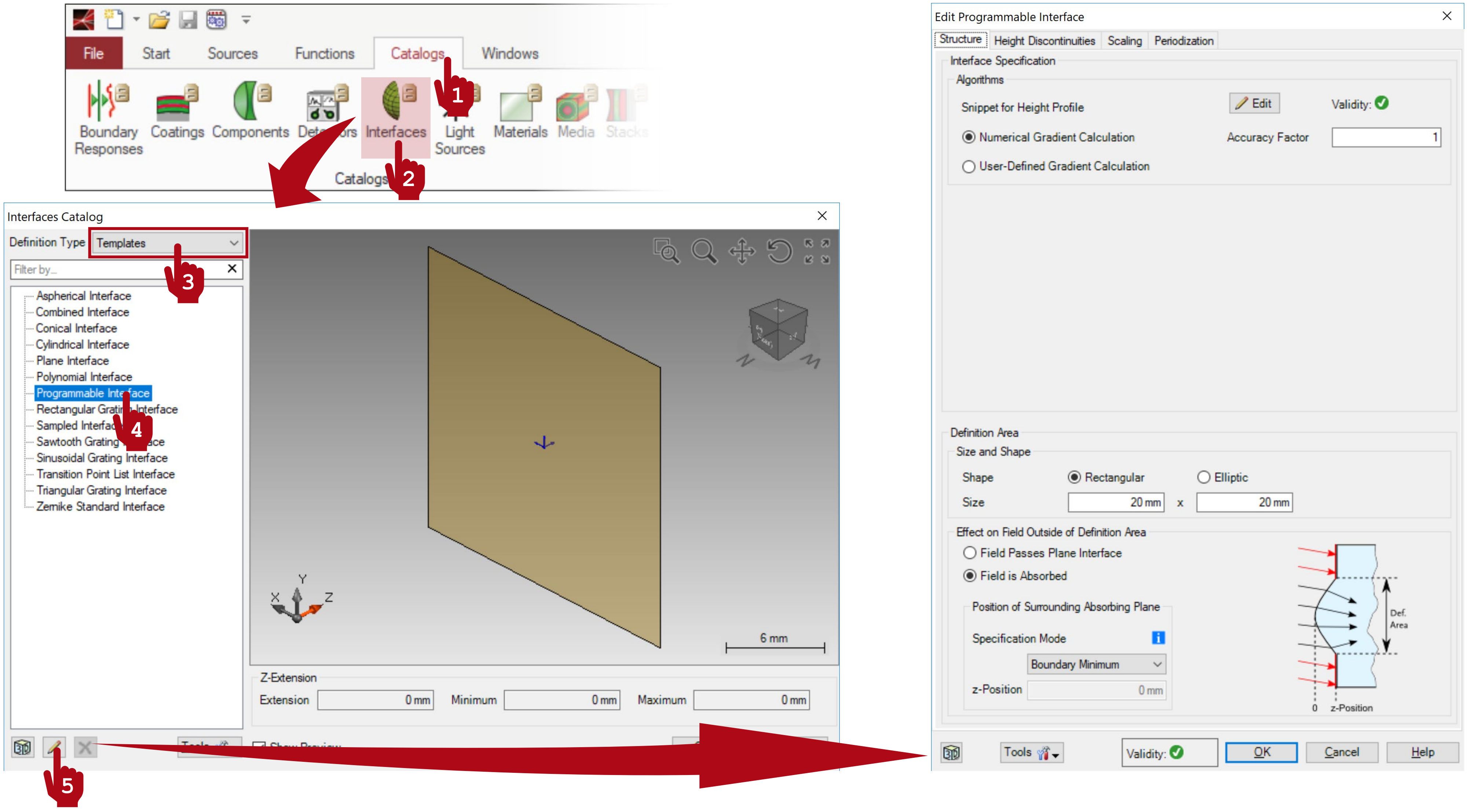
Then, to transform Eq. (1) into an expression that generates the height function *h(x, y)* of a spherical surface

$$h\left(x, y\right) = \pm\sqrt{R^2 - \left(x^2 + y^2\right)} \,. \tag{2}$$

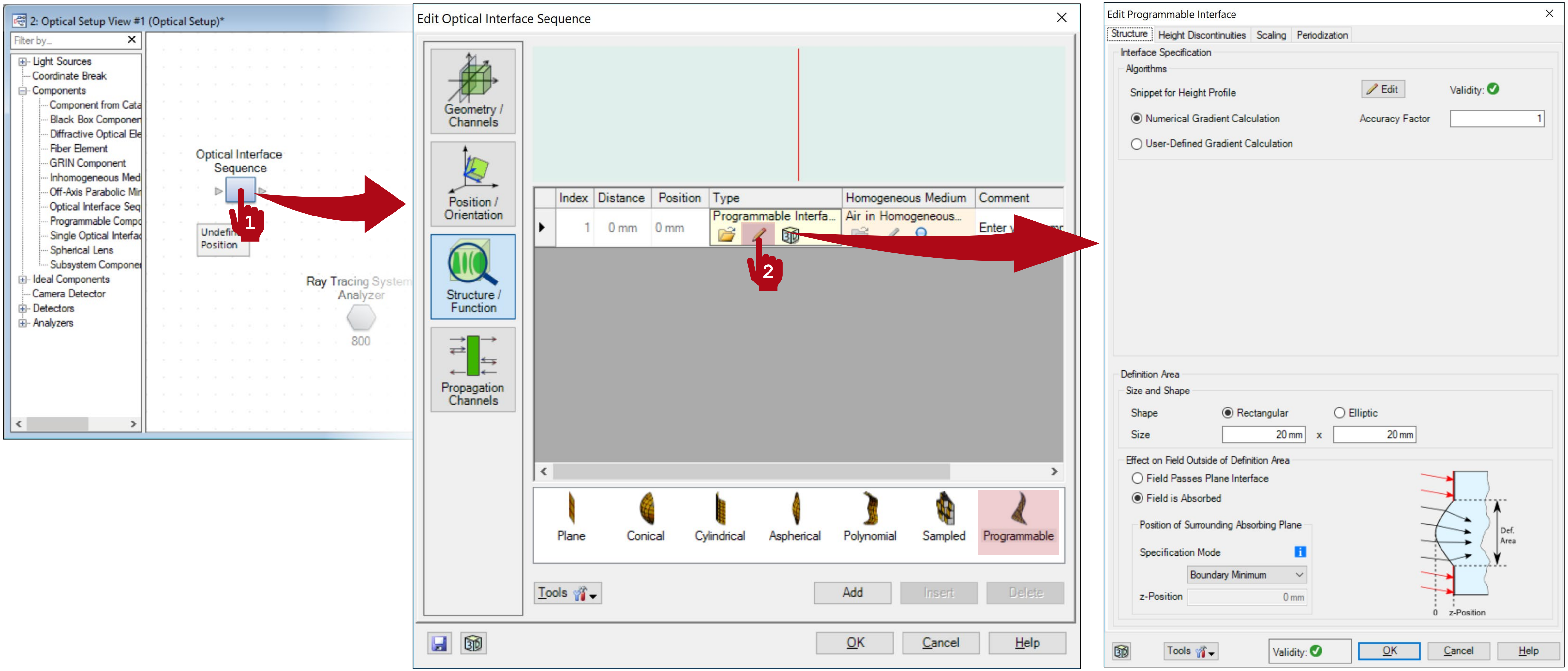
It is also possible to straight-forwardly compute the gradient of the surface

$$\frac{\partial h\left(x, y\right)}{\partial x} = \mp x \left[R^2 - \left(x^2 + y^2\right)\right]^{-1/2} \text{ and } \frac{\partial h\left(x, y\right)}{\partial y} = \mp y \left[R^2 - \left(x^2 + y^2\right)\right]^{-1/2} \tag{3}$$

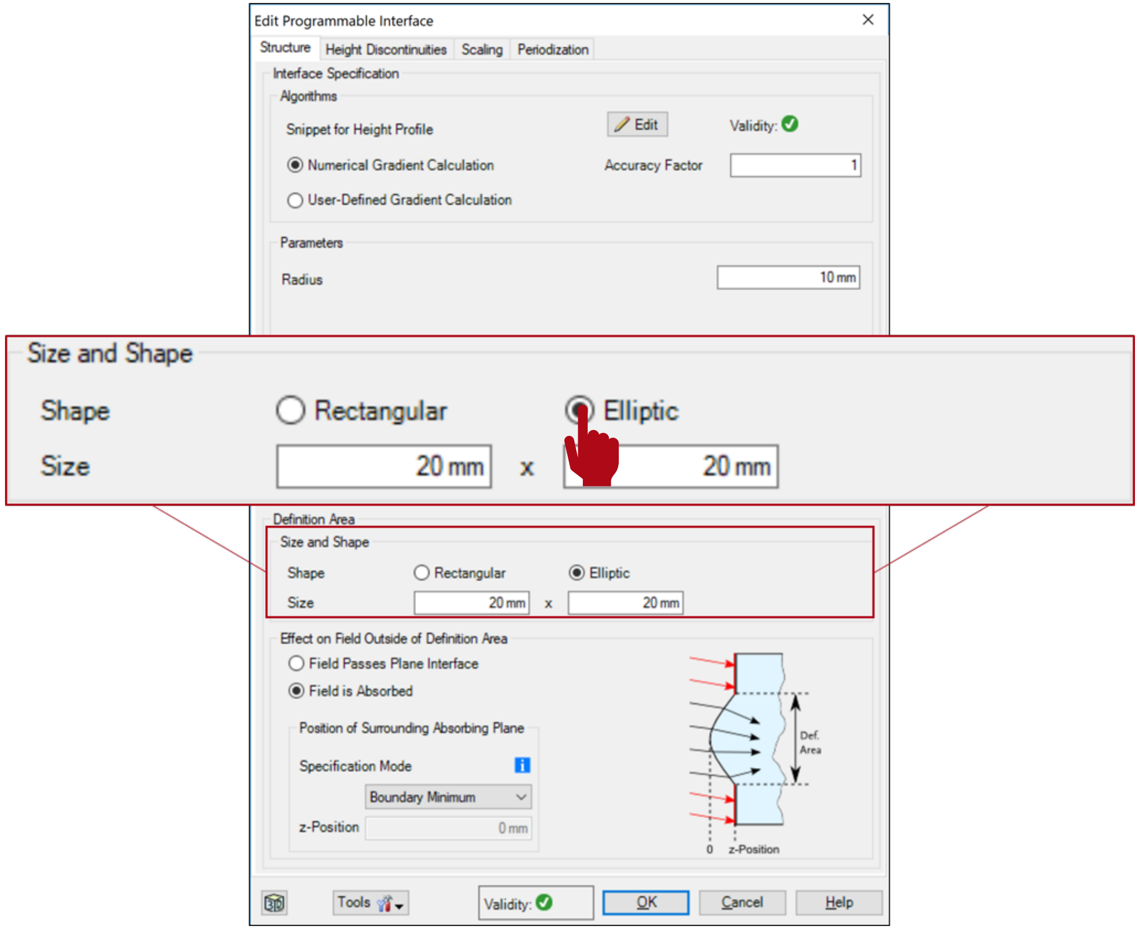
Where to Find the Programmable Interface: Catalog



Where to Find the Programmable Interface: Components

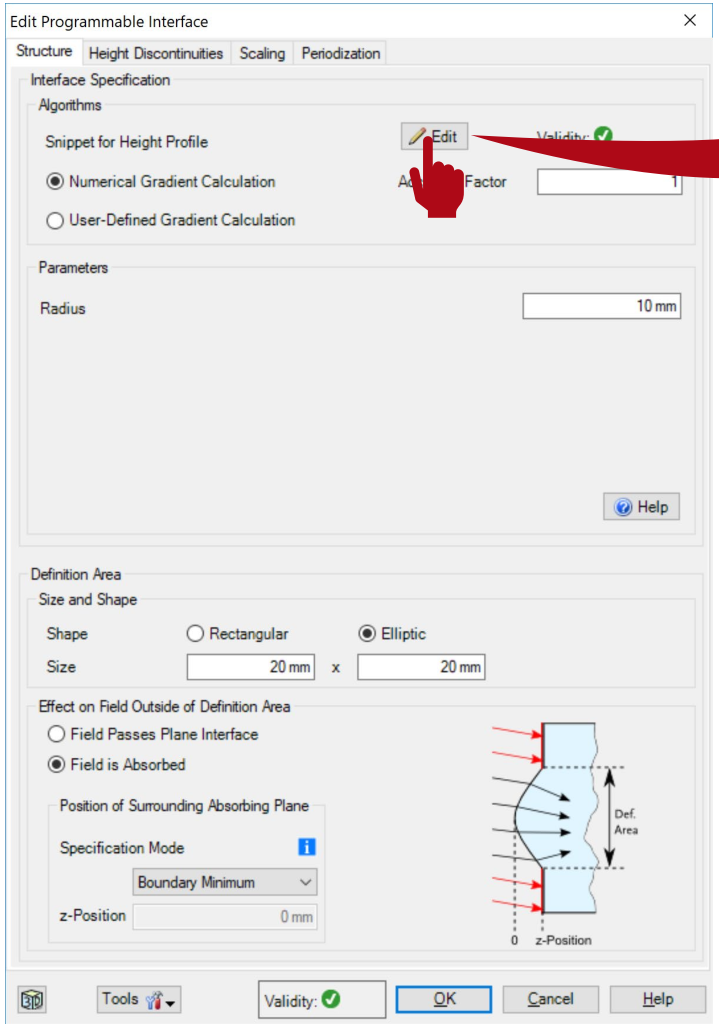


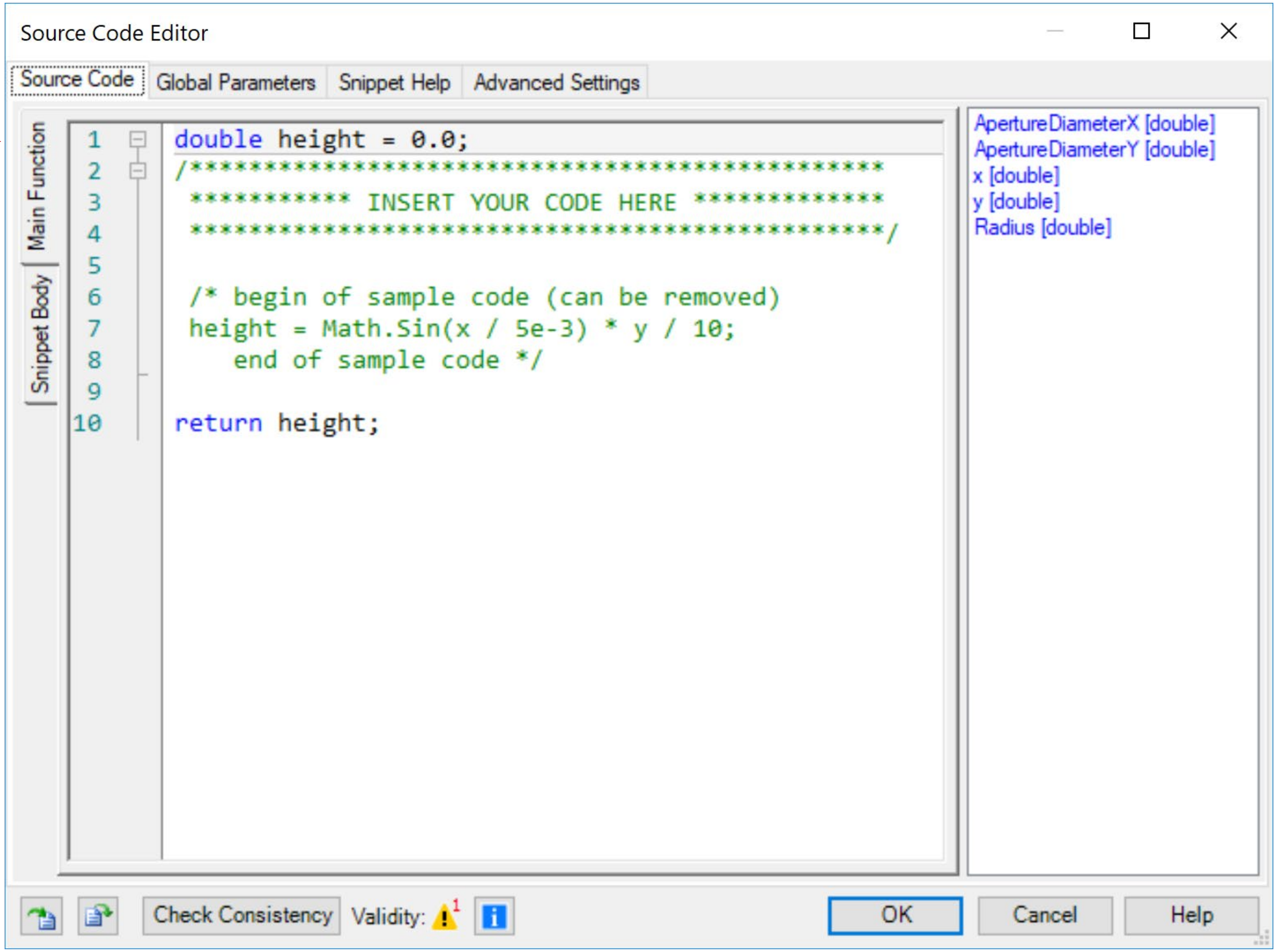
Setting Up the Area of Definition of the Surface



- In the configuration dialog previous to entering the programming interface, the user can define the shape and size of the area of definition (parametrization) of the surface.
- In this example we use a circular aperture.
- Bear in mind that there is a natural limit to the area of definition of a spherical surface, given by its diameter, outside of which the surface is not defined!

Entering the Programming Interface





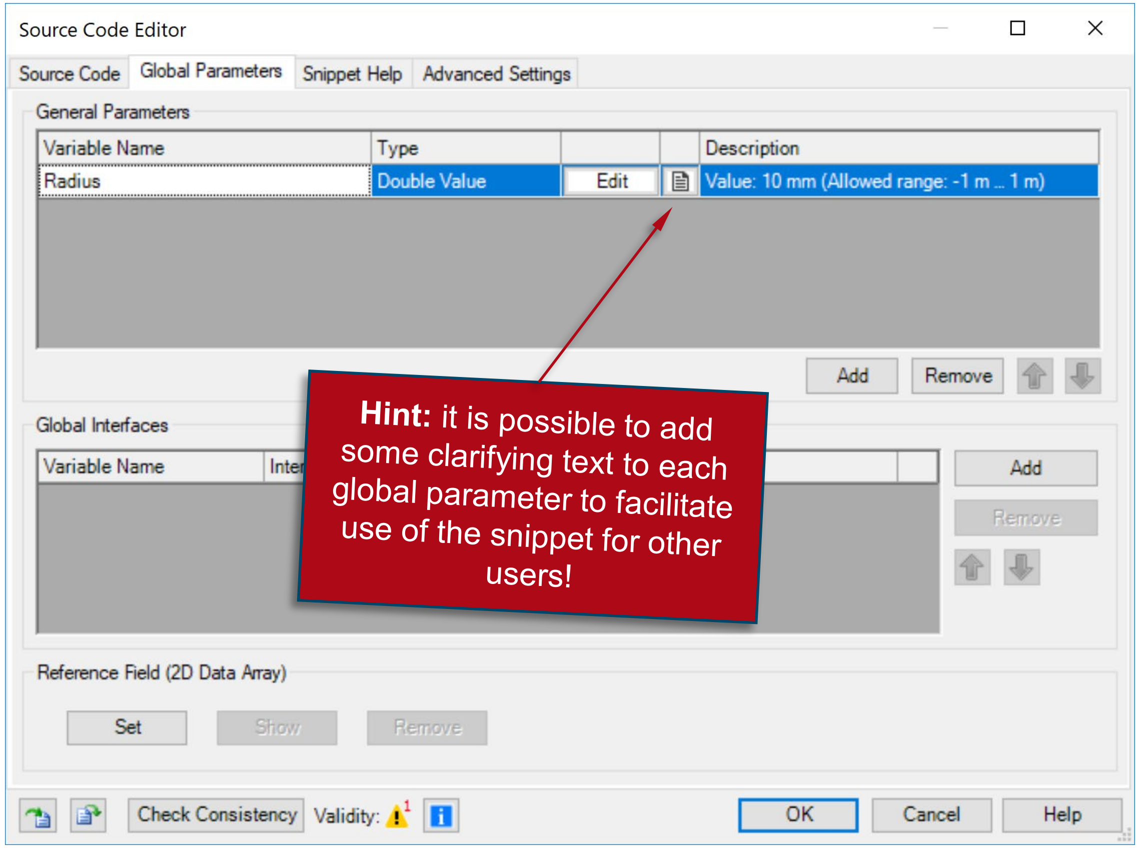
Programmable Interface: Global Parameters

- Once you have triggered open the Edit dialog (Source Code Editor), go to the Global Parameters tab.

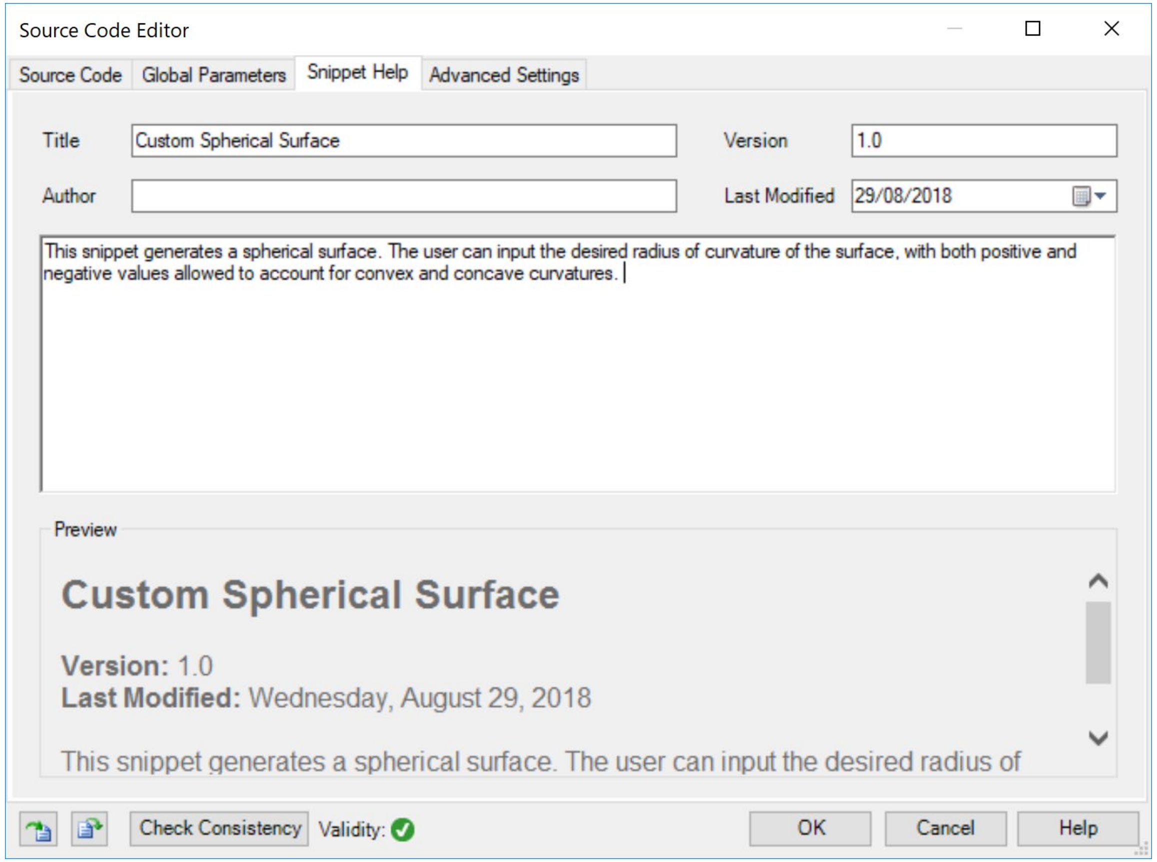
- There, Add and Edit one parameter:

  - `double Radius = 10 mm (-1 m, 1 m)`: the radius of curvature of the spherical surface.

- Use the button with the small "notes" icon to add some explanation to your custom global parameters.

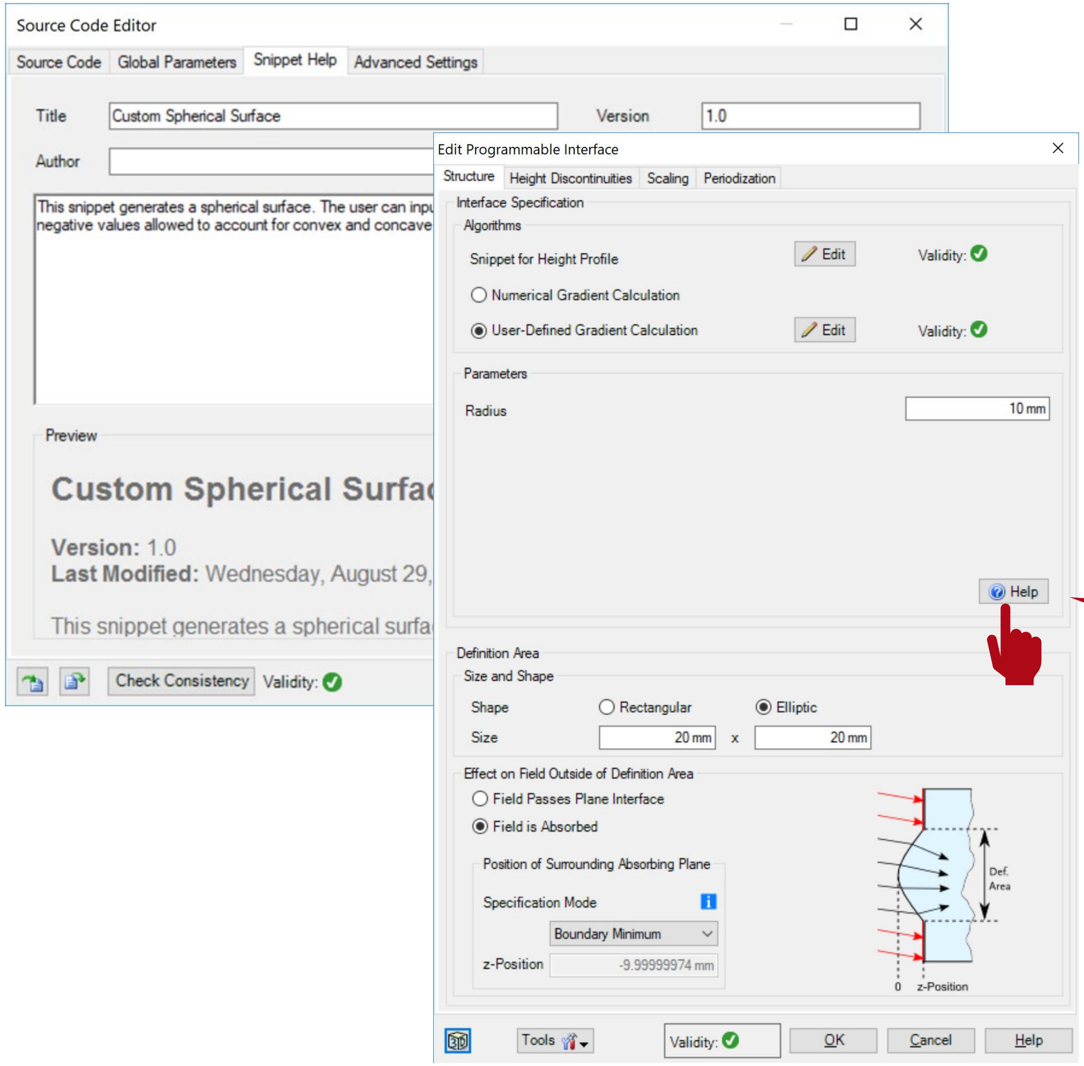


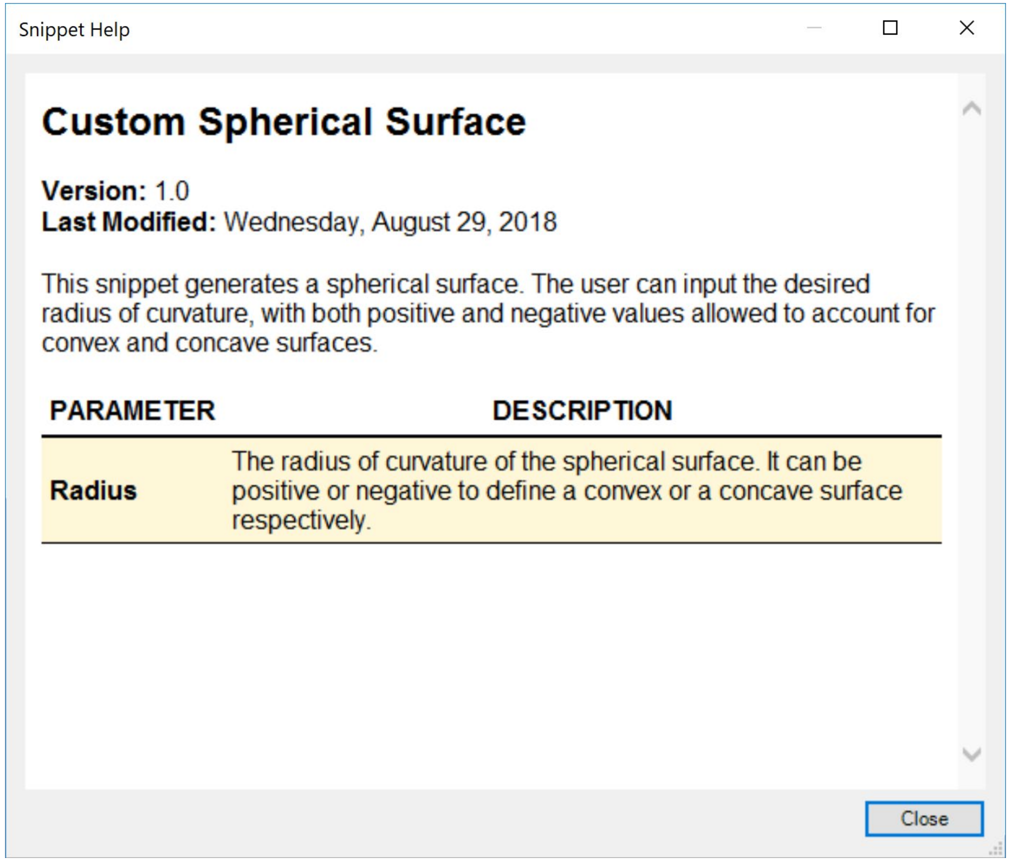
Programmable Interface: Snippet Help



- **Optional:** you can use the Snippet Help tab to write instructions, clarifications, and some additional data associated to your snippet.
- This option is very helpful to keep track of your progress with a programmable element.
- It is especially useful when the programmable element is later disseminated to be handled by other users!

Programmable Interface: Snippet Help



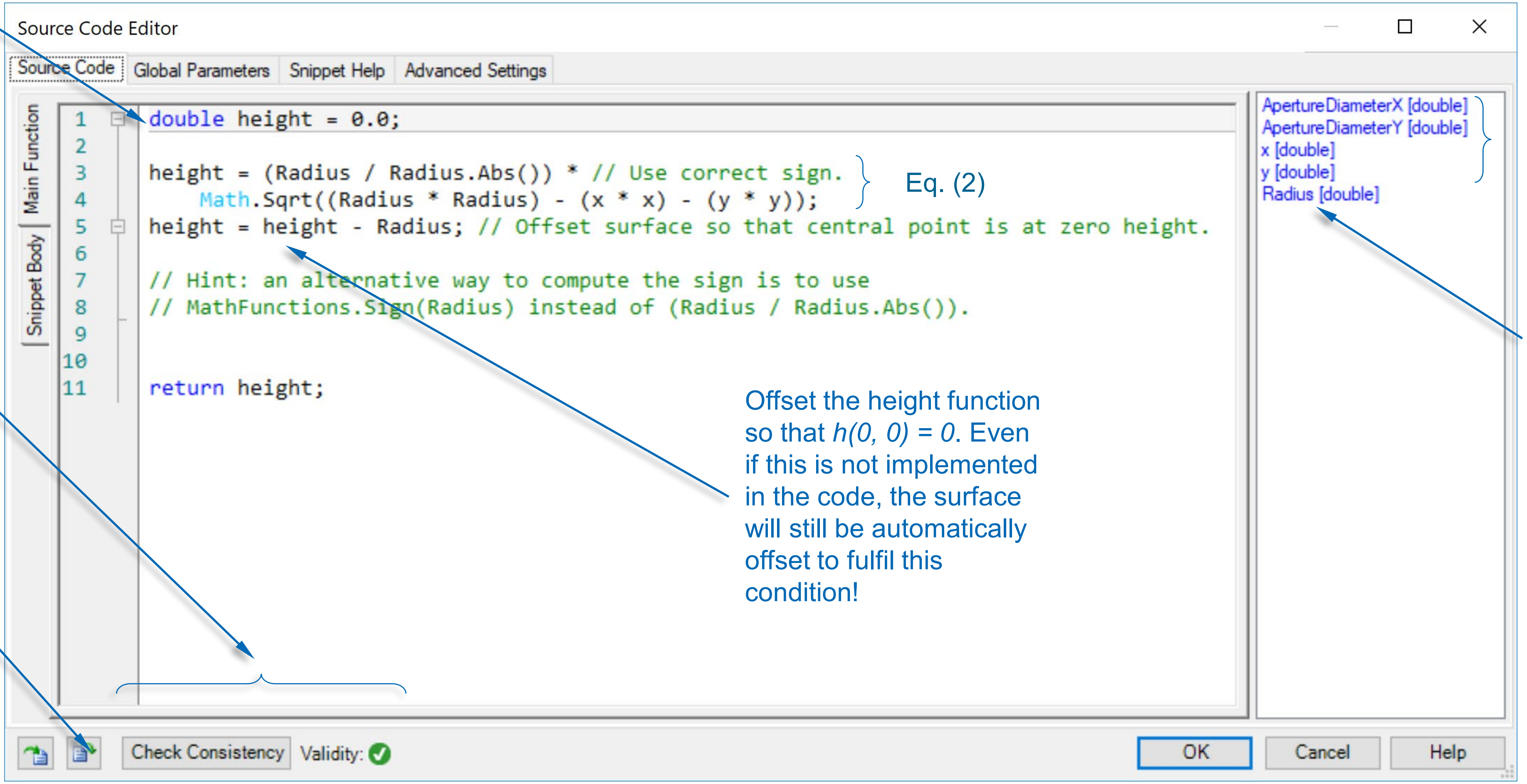


Programmable Interface: Writing the Code

Declaration of output variable given by default

Are there errors in your code?

Export Snippet to save your work!

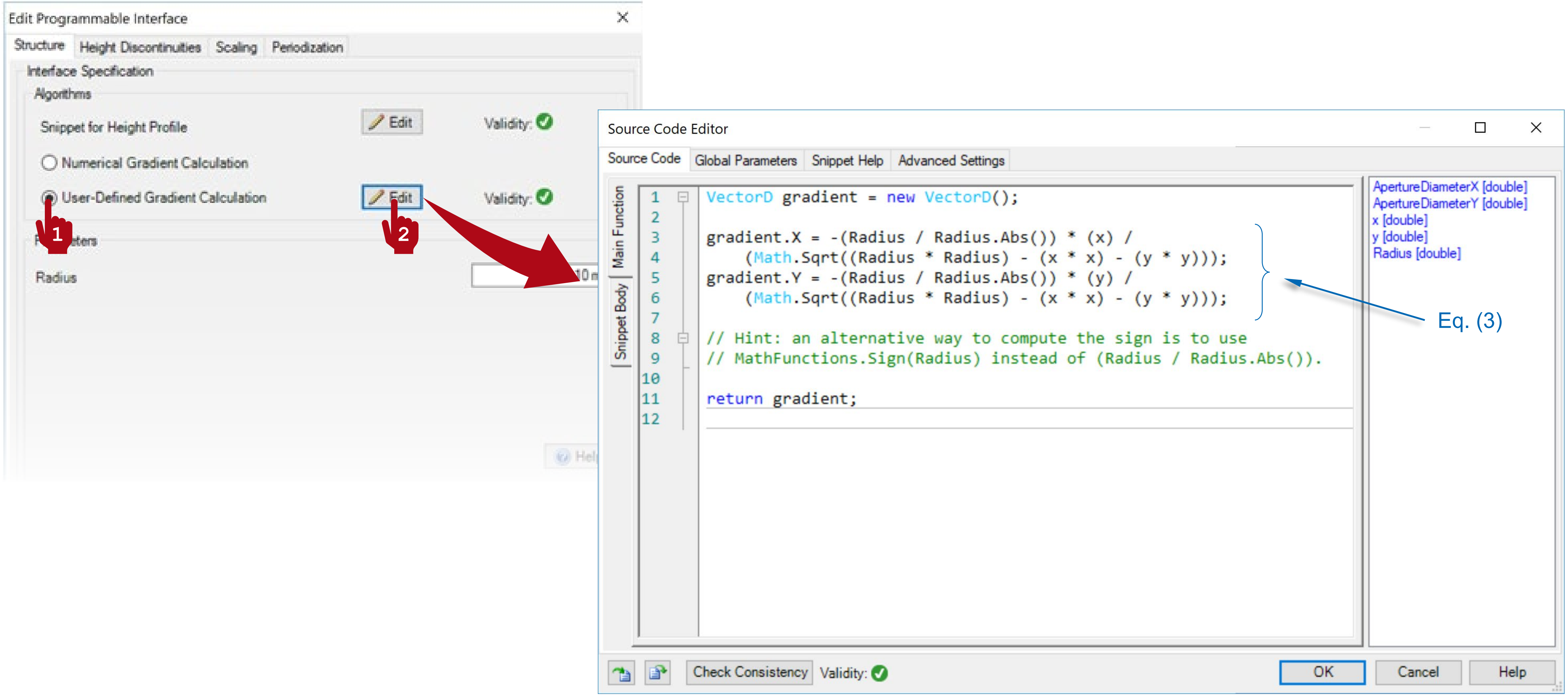


Default global parameters/variables

Global parameter defined by user in Global Parameters tab

Offset the height function so that *h(0, 0) = 0*. Even if this is not implemented in the code, the surface will still be automatically offset to fulfil this condition!

Programmable Interface: User-Defined Gradient

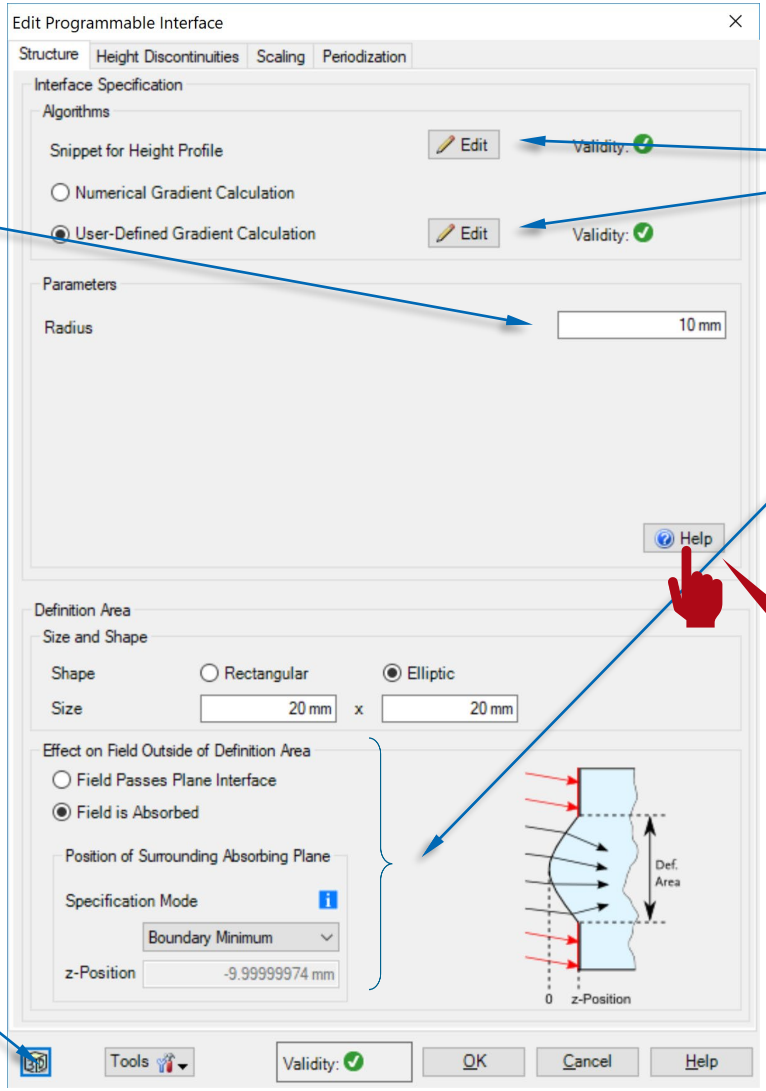


Programmable Interface: Using Your Snippet

You can modify the value of the global parameters you defined here

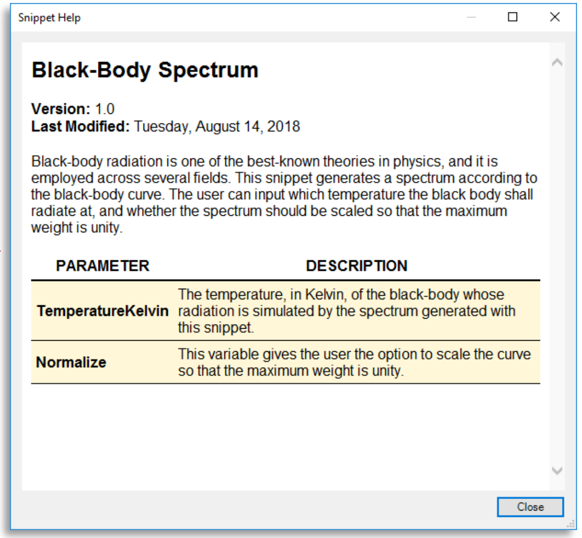
You can modify the area of definition (aperture) of your surface. Bear in mind possible constraints (as already pointed out in the case of the spherical surface).

You can view the surface you have defined here

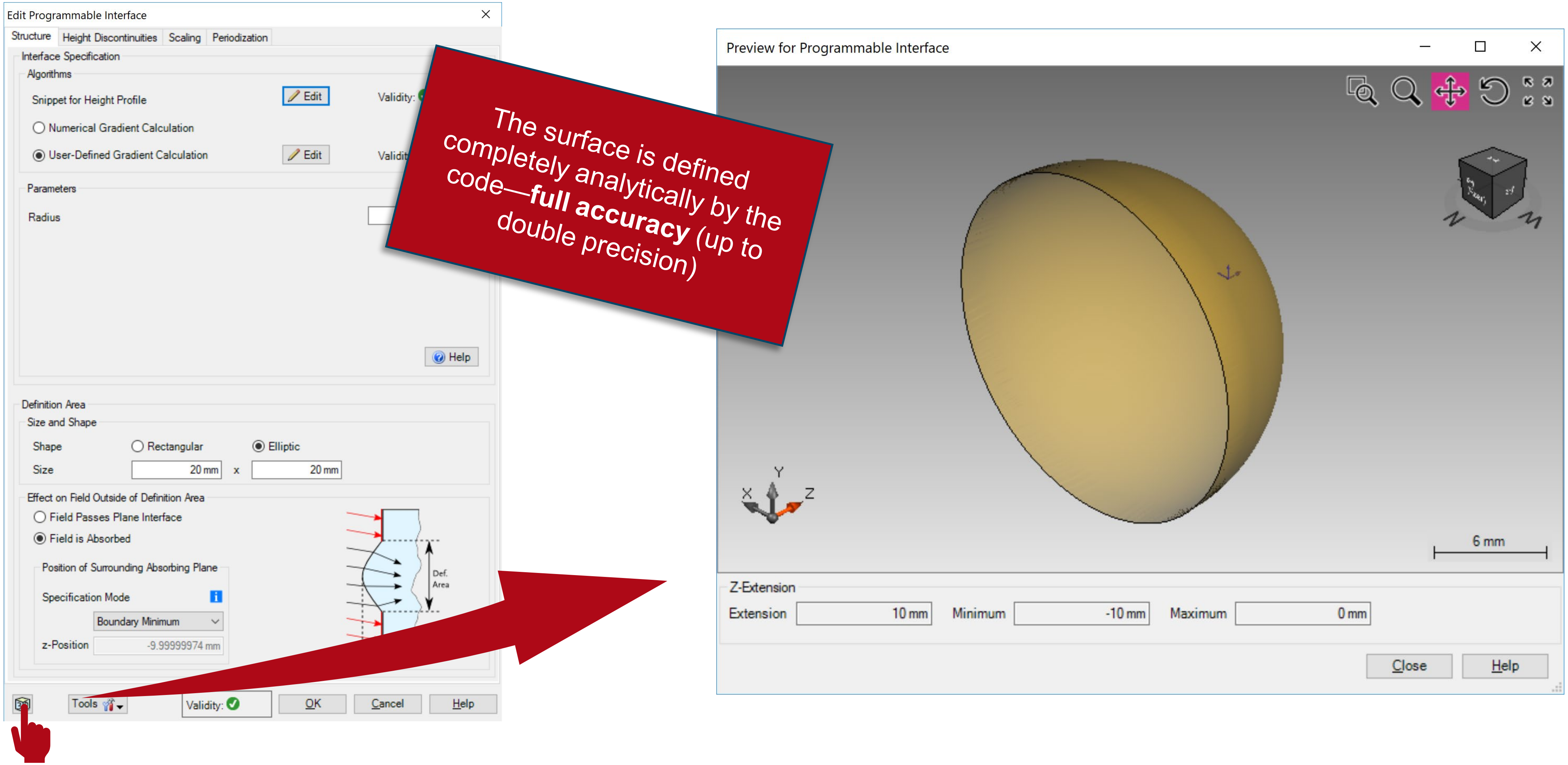


Modify your snippet by clicking on *Edit*

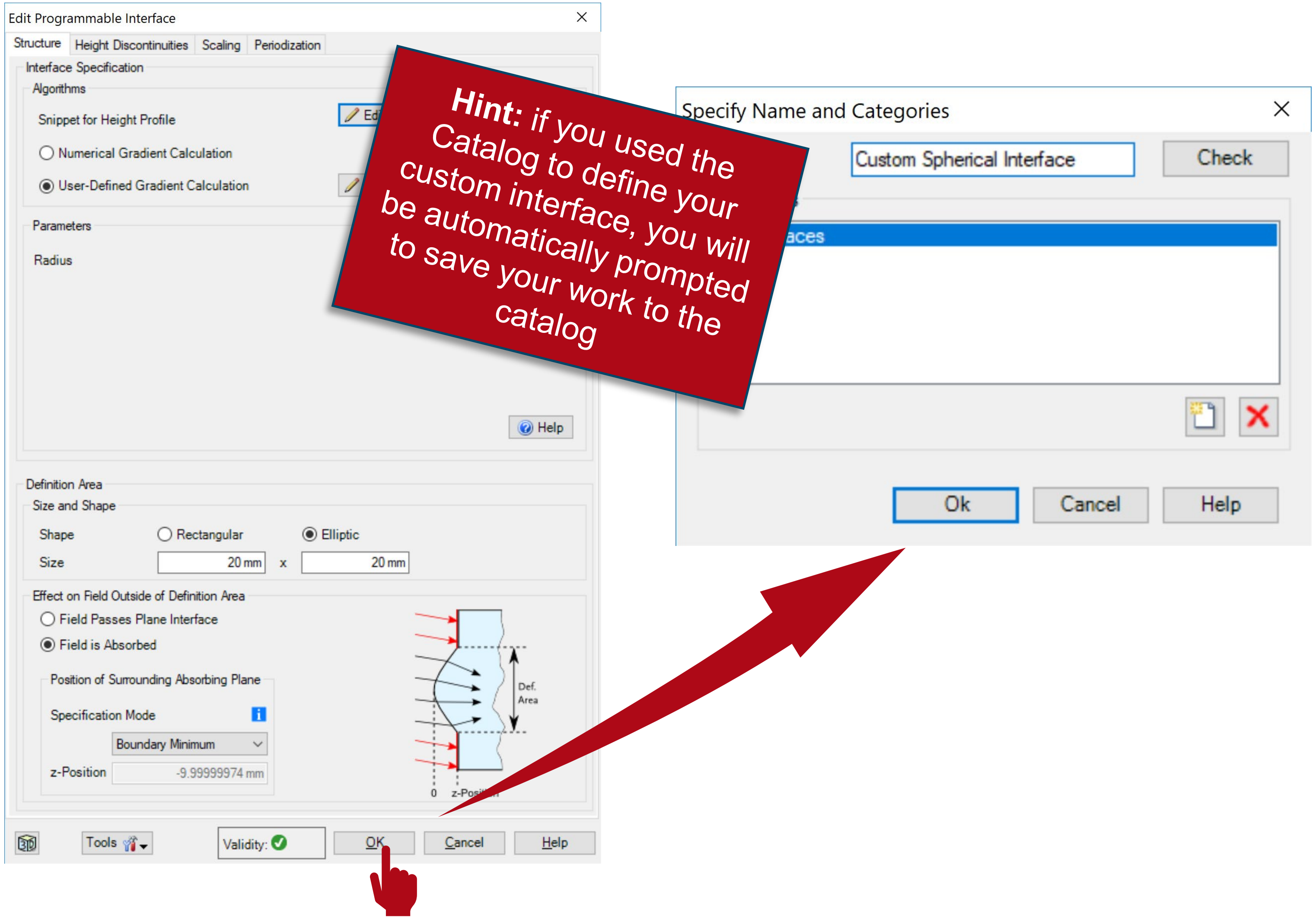
With respect to which reference point in the surface is the surrounding plane defined? And what happens when a part of the EM field meets this surrounding plane?



Output of Programmable Interface

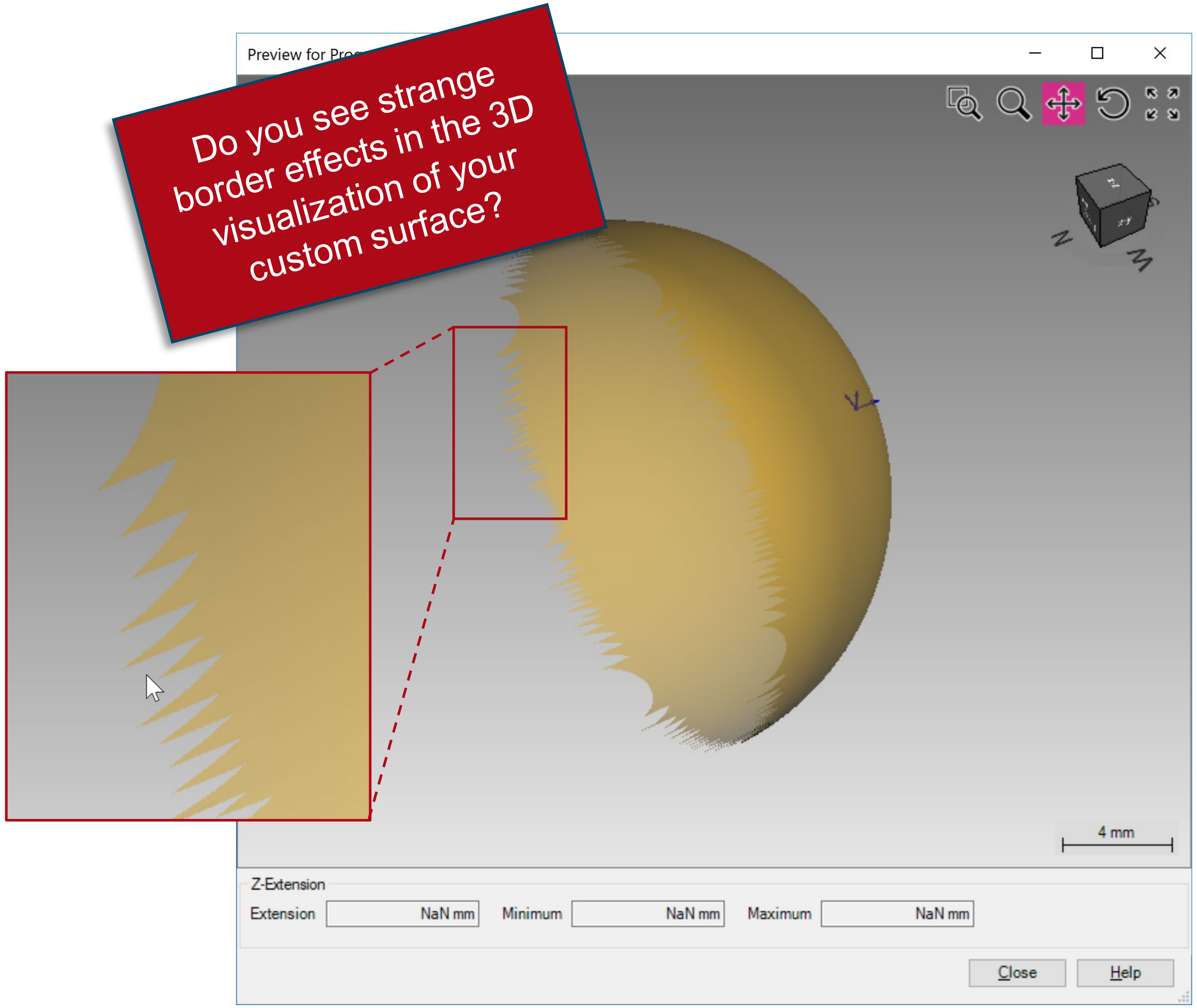


Saving the Custom Interface to the Catalog



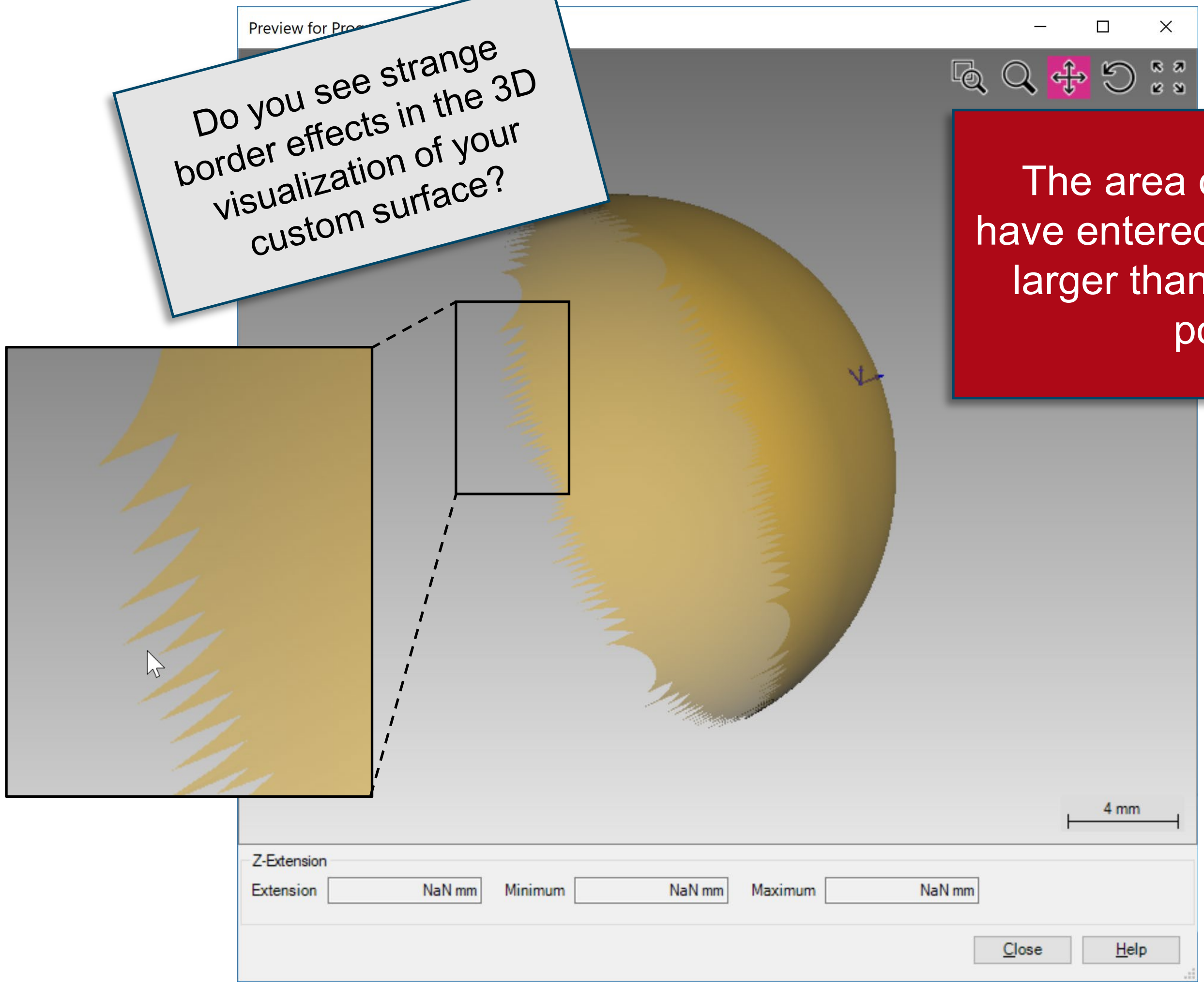
Hint: if you used the Catalog to define your custom interface, you will be automatically prompted to save your work to the catalog

Troubleshooting Tips

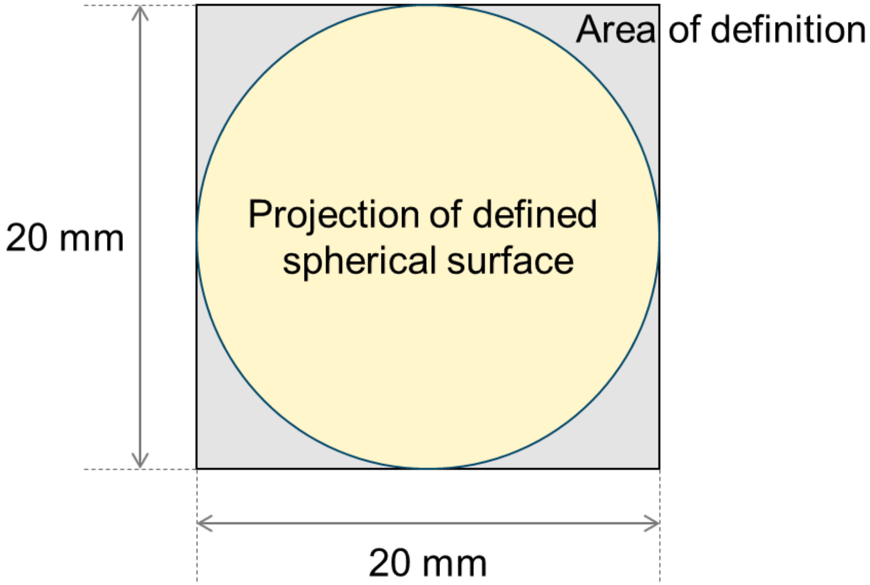


Troubleshooting Tips

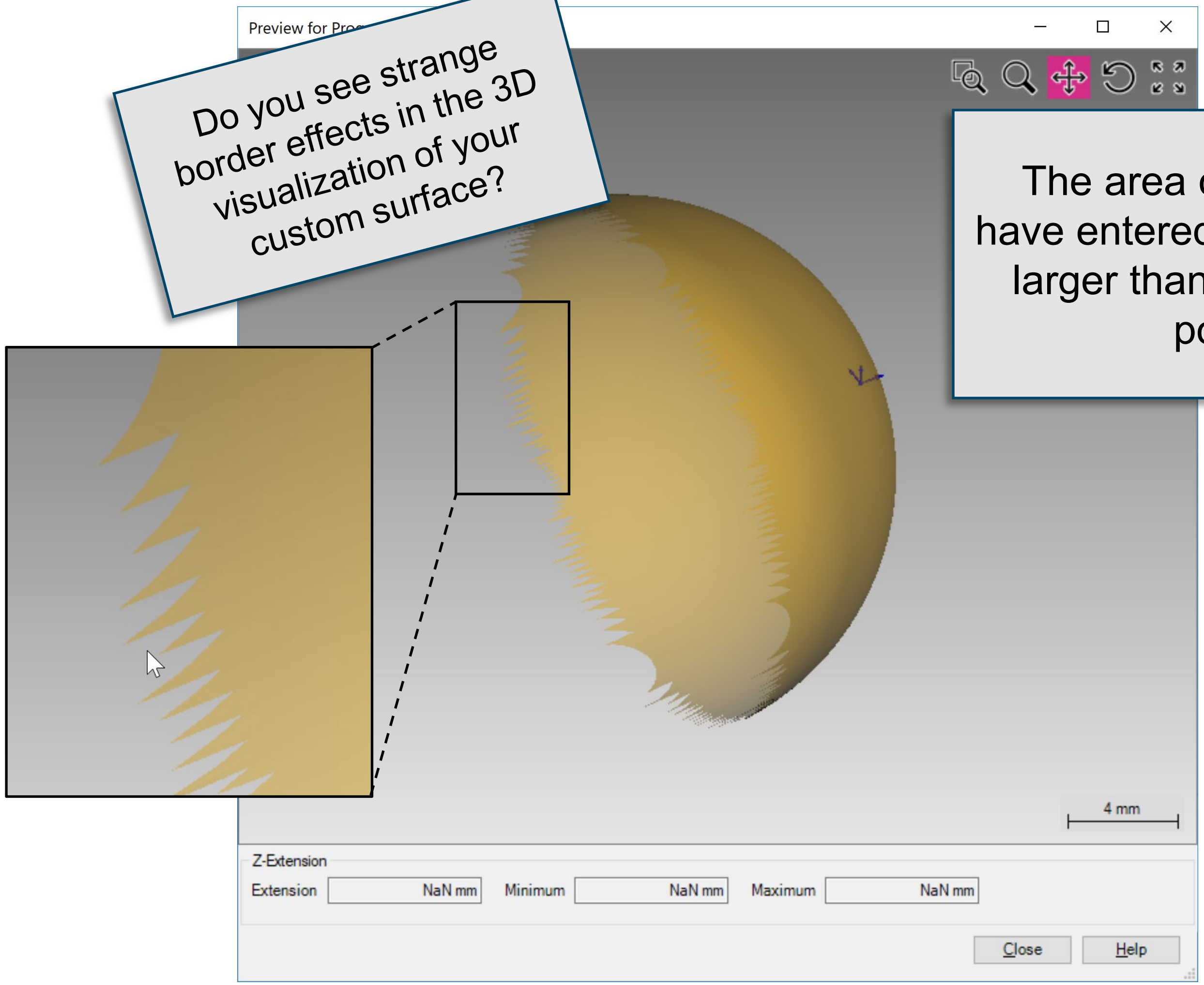
Do you see strange border effects in the 3D visualization of your custom surface?



The area of definition you have entered for the surface is larger than mathematically possible!

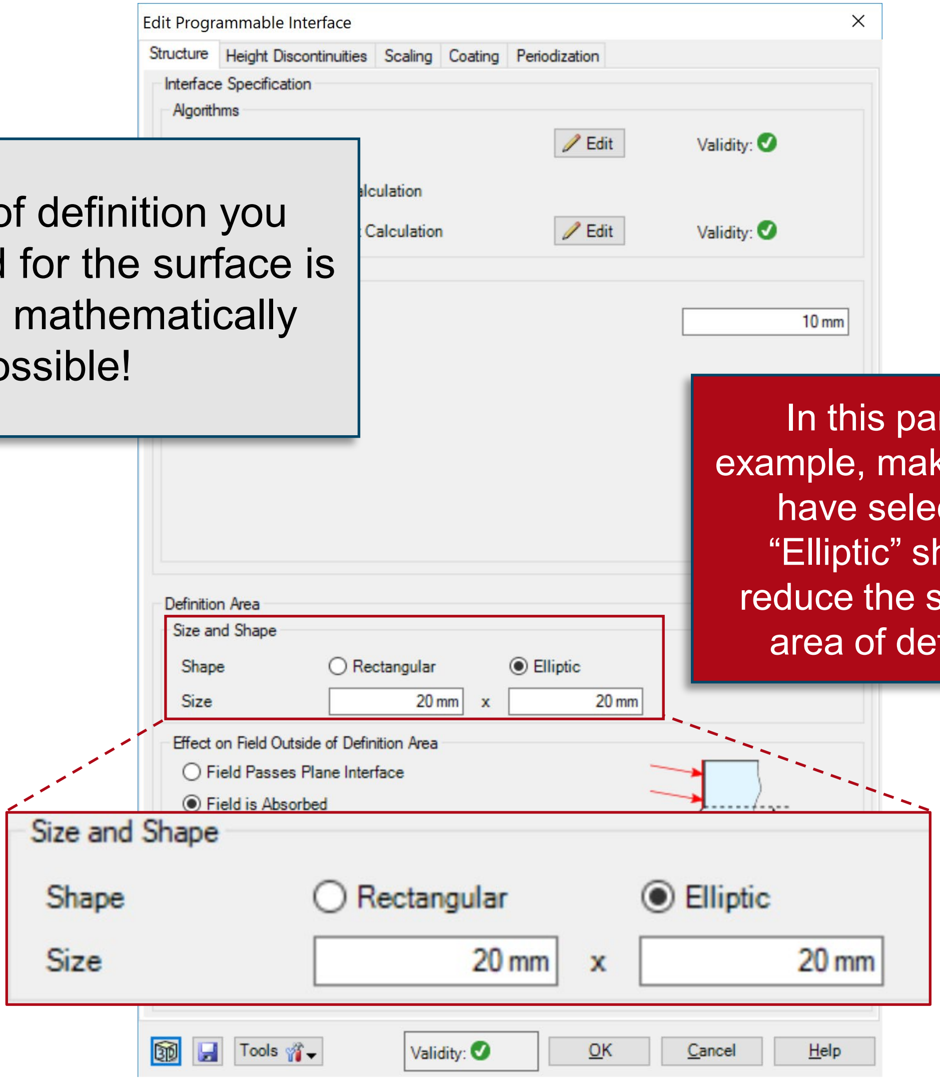


Troubleshooting Tips



Do you see strange border effects in the 3D visualization of your custom surface?

The area of definition you have entered for the surface is larger than mathematically possible!



In this particular example, make sure you have selected an “Elliptic” shape, or reduce the size of the area of definition .

Test the Code!

Main Function (Height Profile)

```
double height = 0.0;

height = (Radius / Radius.Abs()) * // Use correct sign.
    Math.Sqrt((Radius * Radius) - (x * x) - (y * y));
height = height - Radius; // Offset surface so that central point is at zero height.

// Hint: an alternative way to compute the sign is to use
// MathFunctions.Sign(Radius) instead of (Radius / Radius.Abs()).


return height;
```

Test the Code!

Main Function (Gradient)

```
VectorD gradient = new VectorD();

gradient.X = -(Radius / Radius.Abs()) * (x) /
    (Math.Sqrt((Radius * Radius) - (x * x) - (y * y)));
gradient.Y = -(Radius / Radius.Abs()) * (y) /
    (Math.Sqrt((Radius * Radius) - (x * x) - (y * y)));

// Hint: an alternative way to compute the sign is to use
// MathFunctions.Sign(Radius) instead of (Radius / Radius.Abs()).

return gradient;
```

Document Information

title	How to Work with the Programmable Interface & Example (Spherical Surface)
document code	CZT.0096
version	1.0
toolbox(es)	Starter Toolbox
VL version used for simulations	7.4.0.49
category	Feature Use Case
further reading	- Customizable Help for Programmable Elements - Programming a Sinusoidal Surface