ReDim (dynamically allocating memory)

If at the beginning of the programme you are not quite sure yet what size your array will have, maybe because you ask the user in the programme somewhere to specify a value, one can leave the seize open during declaration and dimension the array later. The is called dynamically allocating memory and is syntactically defined by the *ReDim* statement:

```
Dim points() As Double
...
ReDim points(total - 1) As Double
```

If one wants to leave the dimensioning of the array open, the brackets following the array name must the empty like in the example above. In the hyper_surface code example, the points array contains all coordinates of the mesh. The value of total-1 represents the total amount of coordinates.

Flags

Flags are a programming technique that signals to the code what to do. Flags are always variables, because they can have different states. In the hyper_surface code example, we use the boolean variable outside as a fl ag. The fl ag can be True or False and thus tells the code what part to execute next.
One can think of a fl ag as a traffi c light, which when one of three colours indicates to the driver how to approach the intersection.

Boolean Variable Type

The most basic flag is the TRUE or FALSE boolean expression. If you only want to set a control variable to indicate if something is TRUE or FALSE, you can declare the variable to be of boolean type. In the dynsur() procedure we declare:

```
Dim outside As Boolean
```

Thus, outside can only become TRUE or FALSE.

You can use TRUE and FALSE also as a numerical operator, where FALSE = 0 and TRUE = -1.

In conditional statements, like *If(condition = True)Then...* or *Do...Loop While(condition = True)*, a boolean variable can be used as a flag. If the boolean variable in the condition expression needs to be tested for True, one simply writes:

```
If(outside)Then ... , instead of
If(outside = True)Then ...
```

Otherwise, if the conditional expression were to be tested for False, one writes:

```
If(Not outside)Then ..., instead of
If(outside = False)Then ...
```

That is because the variable outside contains the value of True or False anyway.

Do...Loop While(conditional statement)

The only loop you have come to know so far is the *For...Next* loop. In the *For...Next* loop you can iterate only according to a fixed number of cycles. If you want to iterate through a loop as many times as necessary until a condition is reached - like combining a loop with an *If-statement* - then you will have to use the *Do... Loop While(condition)*.

In the hyper_surface code example in the dyn-sur() procedure, the loop repeats as long as none of the coordinates of the mesh reach the *max* or *min*. If any coordinate, tested through *verti()*, goes beyond this limit, the boolean variable *outside* will be set to True and thus, the condition for the loop is not fulfilled any longer, which stops the loop:

```
Do
      ...
Loop While (Not outside)
```

Logical Operator *Or / And*

A logical operator determines the result of a comparison between two expressions. The result then tells the program how to proceed. In the hyper_surface code we use the ***Or*** operator to test if either of the coordinates is outside of the set limits max *OR* min:

```
If ((verti(0) > max Or verti(0) < min) Or _
    (verti(1) > max Or verti(1) < min) Or _
    (verti(2) > max Or verti(2) < min)) Then outside = True
```

As you can see, just one of the conditions has to occur then outside will be set to *True*. ***Or*** checks if only one of the two conditions is true the expression will be True, whereas ***And*** requires both conditions to be true for the expression to evaluate to True.

An example can illustrate this:

```
If(wheather = rain Or wheather = cold) then stay in
```

If it only rains I stay in. If it's only cold I stay in. Even if it rains and it is warm, I stay in. Even if it's cold and the sun is shining, I stay in.

But if we replaced the **Or** with an **And** there would be only one possibility:

```
If(wheather = rain And wheather = cold) then stay in
```

Only if it rains *and* it is cold, I stay in. I have to go out if it's sunny but cold or if it's raining but warm...clear?

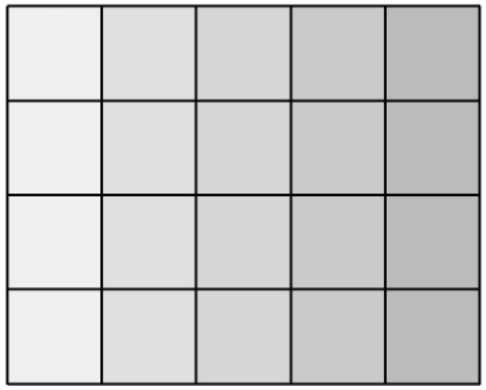
safety flags indicating actions

Nested Loops

By far the hardest syntax to understand in the hyper_surface code example is the nested loop and what it does. Nested loops are like a Russian doll - one inside the other, inside another,....

Just like with Russian dolls where you have to close the smaller/ inner dolls before closing the bigger/ outer ones, you start to open the outer loops, then open the next smaller loop. In order to close the outer loops, you have to close the inner one first.

In the code example, we want to lay out the mesh coordinates according to the indeces of the loops in order to get a rectangular symmetric structure.
Msize are the X-axis coordinates and the index of the outer loop (For *n*). Nsize are the Y-axis coordinates and the indeces of the inner loop (For *m*).
Notice that when iterating through the loops, between every m-index, we do all the n-indeces. That way, we always close the For n loop first before adding another n-index.
That is represented by the shading columns in the matrix image. Fill in the fields with their m-size and n-size indeces!



Show

Go check out the exhibition called *The Algorithmic Revolution* at the *Zentrum fuer Kunst und Medium* (ZKM) in Karlsruhe, Germany. It's well worth a visit during your long Christmas break.