Why program at all?

Over the last two weeks we looked at mother nature's and a mechanical way of organizing elements within a systems through local feedback to achieve a global phenomenon.
The soap-film's molecules, aka mother nature, knew nothing of the overall shape that emerged through that specific organization - when we all went: oah nice, wow like a tent, etc. Neither did the molecules know they were part of the surface. All they knew was that about their 'personal' relationship to their surrounding's. That relationship between each other displaced them for which from our perspective beautiful surfaces were generated.
One of the key notions is that the molecules had no knowledge about the simultaneity with which the adaptive feedback processes between them happened.

Nowadays, computers have reached enough speed to simulate such simultaneous or parallel processes (although, as we know already from NetLogo, that parallelism is fake) in real-time. Additionally, sciences like biology, neuroscience, chemistry, psychology, system theory and others have explored natural phenomena that rest on such parallel processes. Infact, it seems that all complex organizations or systems use distributed parallel processing.
Another way of describing such processes is to say that local information is computed simultaneously!
In order to embed design mechanisms and architectural concepts into their contextual systems, we explore mechanisms from other fields and try to synthezise them with hitherto well working methods of design. Thus, we might be able - thanks to the capacity of computational simulation - to adapt some architectural design strategies to the underlying complexity of our world.

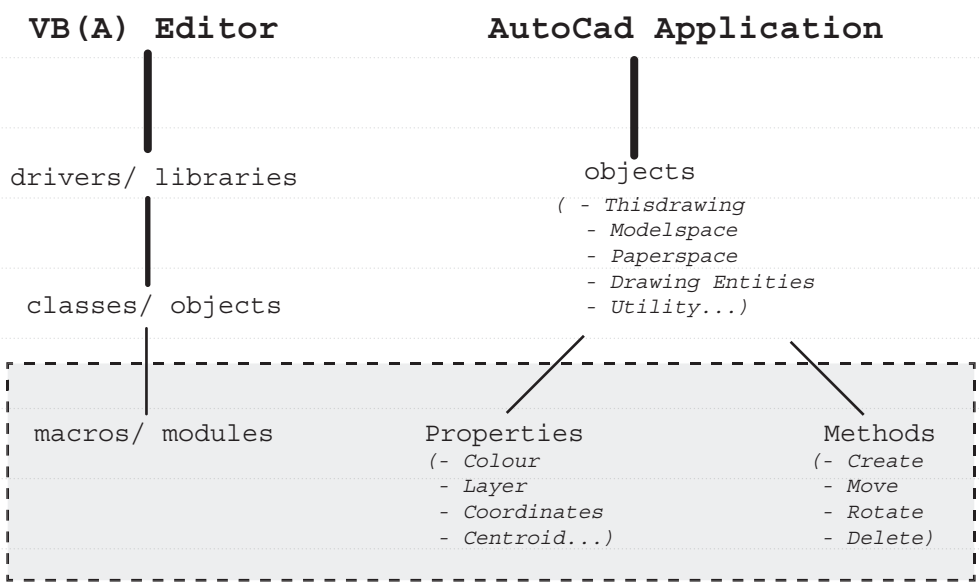
The VBA AutoCad environment

Today we will demonstrate to you live how to write a little program in Visual Basic for Applications. The application in that context is AutoCad 2002. The difference between Visual Basic and Visual Basic for Applications is that VBA needs to be embedded in an application that has a 'front engine' to output the compiled code. VB on the other hand has its own interface and output, which builds the compiled code into stand alone applications.
We will be using VBA AutoCad so that we don't have to write our own graphic output interfaces, because we will be using the application's, here AutoCad. Therefore, you can just tell the application to draw a cube and collect the information in a data-structure. On the next page, there is a little description diagram of the AutoCad object.

In order for you to write an algorithm into VBA AutoCad, the environment should be a little familiar to you:

1 - to make a new project, you need to open the VBA manager (get in the *Tools* menu, *Macro* tab; or simple type *vbaman* into the *command line*) and chose *New*. A new *ACADProject* will appear, which will be called *Global1*. Now press *Visual Basic Editor* and you will find yourself in the editor/ compiler.

2 - one can write an algorithm directly into the current drawing, a new sheet called *ThisDrawing* diplayed in the *Project* window on the left, or insert a module to write code into. We will write into modules since they can be mixed and linked with other modules, classes and forms to create larger projects. So, right-click in the *Project* window and chose *Insert* a *Module*. Here you will write your little program of the day later.



3 - before doing so, give a name to your project by typing a name without spaces or numbers into the name field of the *Properties* window on the left just underneath the *Project* window.
To save your module you can simple save it in the editor. But if you were to save it as another, one has to go through the *VBA Manager* again.

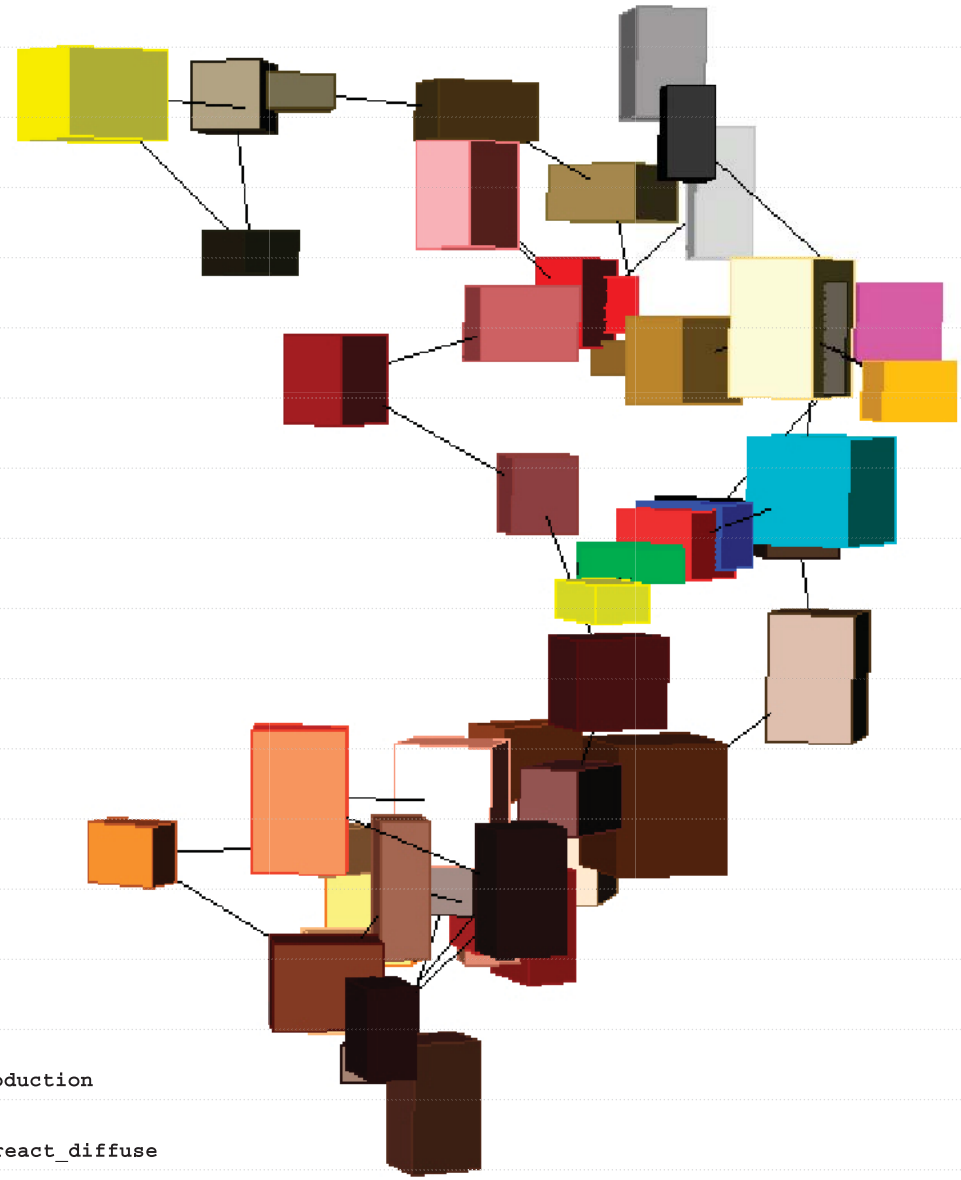
4 - later, once you have written the program we wrote during the class, you will want to run the project. First though, you will have to compile your algorithm. Go the the *Debug* menu and chose the first tab called *Compile ACAD-Project*. If no compile errors are found, you can do two things:

a) go back to AutoCad and type *vbarun* on the *command line*, which brings up a dia log where you can chose your project and then run it.
b) just press *Run* in the editor. To see the program unfold in the AutoCad inter face though, you will need to make the editor really small first, so as just to see the *Run* button.

5 - having run the algorithm and being generally happy about the outcome, you should *Unload* your project in the VBA Manager. If you want to access and run you project at a later stage again, you will have to *Load* it through the VBA Manager. One can load multiple projects.

TASK

Change the program and parameters. For other objects to use instead of Box and Sphere, look up the Developer Help in AutoCad. It is very thorough, so you don't have to buy a book!



```
'****************************** today's line_walker program ************************

Sub main()

    Dim i As Integer
    Dim here(2) As Double, there(2) As Double
    Dim walker As AcadLine
    Dim node As Acad3DSolid

    there(0) = random(0, 25): there(1) = random(0, 25): there(2) = random(0, 25)

    For i = 0 To 50

        here(0) = there(0): here(1) = there(1): here(2) = there(2)
        there(0) = random(there(0) - 15, there(0) + 15)
        there(1) = random(there(1) - 15, there(1) + 15)
        there(2) = random(there(2) - 15, there(2) + 15)

        leng = random(3, 8): wid = random(3, 8): high = random(2, 10)

        Set walker = ThisDrawing.ModelSpace.AddLine(here, there)
        Set node = ThisDrawing.ModelSpace.AddBox(here, leng, wid, high)
        node.Rotate here, random(0, 360)
        node.Color = i

        ThisDrawing.Regen (acActiveViewport)
        ZoomExtents

    Next i

End Sub

Function random(low As Double, up As Double) As Double

    random = (up - low) * Rnd + low

End Function
```