Digital Craft Lab: A Case Study

A study about contemporary form-finding methods and how to teach them

by Peter Varo ©2015 for Moholy-Nagy University of Art and Design, Budapest

---

TOC

The problem

If we take a closer look, on how the shapes, textures and form-finding processes changed in the last ten years in the field of product design and architecture, we have to come up with the conclusion, that the rapid evolution of generative and algorithmic geometry creation has slowed down. We could almost say: it has stopped. As a matter of fact, most of the methods and directions we have achieved a decade ago are repeating themselves, and unfortunately not as educated *quotes*, but as *full* copies.

What is the problem then? Are we truly out of ideas or inspirations? Or, are the members of the current generations are not *that* talented? Or, did we hit some kind of *wall* where we have to admit, there is nothing new anymore, we can only repeat ourselves?

The answer for those questions is not a simple one, actually it is rather complex. To find the answer, first we have to take a look at why there was such a rapid evolution ten years ago.

Man, the object-maker

There are several things, why we become the *ruler* of the earth among all the species. One of them *(maybe even the most important one?)* is that we started using and designing objects for ourselves. For instance, we are not as fast as most of the animals, but we can build things which supports us and extends our capabilities, so we can be faster than any of them.

We started designing objects to design other ones. And this is probably the most exciting element of the evolution of our environment. At this very moment the most complex product we are using, are the general purpose computers. Actually that was the point, where the limitations of our

imaginations finally discovered a new tool, which is capable of not only *creating* inspirational materials, but even produce final products for us.

How is that even possible? Let's think about it: if our imagination have a limit (which we can all agree on), then the computer, which was designed by our limited imagination also has to have limitations as well, hasn't it? Well, the truth about computers is such an easy one: these machines are extremely limited and dumb ones. What they can do right now, is approximately at the complexity level of a five years old child.

If that is true, how could they become our inspirations? How can we call them our new muses and/or tools? Computers are extremely good at only one thing: they are *blazingly* fast. Even if they can only do a handful of things, but because they can do those as fast as nothing else can, they are capable of helping us solving the most complex problems we could face with. And that is exactly why they are so important for us, artists and designers.

They never get bored. They never lost interest. They are always precise and they can do jobs at a constant time. They are reliable. But still, they are utterly dumb. And that's exactly why *they still need* us, as creative inputs, which they can process and execute -- as fast as the owners of the original ideas, will never be able to do so.

Basically this is what happened more than fifty years ago. One of the first creative tools has borned, when the first 3D modelling program, the Sketchpad (1963) was created.

Without being too specific and scientific, the whole thing about programming and building complex softwares is about abstraction. How we can abstract the basic states "ON" and "OFF" (has electricity or doesn't) to bits (binary-integers such as 0 and 1); bits to bytes as a chunk in the memory; and those chunks to numbers, characters and operations; these types and methods to functions; the functions to geometry; and geometry to visually representable virtual objects; and these objects to final -- even real life -- products.

This rough and broad abstraction-chain may sounds like an extremely hard thing to build, at least for *ordinary* (not envolved in computer sciences) people. And the truth is: it sure is. Most of the art-oriented people are not really interested in such details of how and why things are working inside a computer. And that was not really a problem in the early days.

Sketchpad was the first one, which was followed by literally hunreds of others in a very short amount of time. Almost all had one main purpose in mind: try to be more abstract, therefore easier to use for end-users. The clients don't have to write complex commands anymore, they are free to use shortcuts. Most of the time they can use their mice, to point and click on icons, which will do the heavy work for them.

This was a great thing at first, since most of the designers wanted to do the exact same tasks over and over again in a 3D modeling application, with different parameters of course. But, when new and complex form-finding ideas came across, like *growing* organic structures based on the same principles as nature solves the similar problems, these predefined tools were not enough.

As a matter of fact, these tools cannot be generalized at all! We may tempted to think, this is a great direction, and we finally arrived at a point, where even the tools of the designers are not the same, and we can literally do *whatever we liked to do*. Unfortunately this is only one side of the truth. The other side is: designers need new skills to extend and create their own tools, so they could research and create more freely. And that skill is programming.

The designer in between art and science

Sadly, these skills are beyond the knowledges of designers. Which also was not a big problem a few years ago, because easy to use tools, such as Grasshopper was created. It is a Visual Programming Language built on top of one of the most widely used NURBS based CAD modeling system called Rhinoceros 3D.

We finally arrived at the point, where the rapid growth started. Tools like Grasshopper and easy to use programming languages such as Processing or MEL helped the experimental designers to create and share encapsulated algorithms, and build their own toolsets. Well, this is also the point where, naturally, this evolution stopped.

Why? Because such *ultra high level* encapsulations have their own limitations as well. It is quite easy to create even complex surface panelings, generate random situations, evolve a system toward attractors or combine these three together, but basically that's all one can do with no to little effort. If a designer wants to think outside the box, one finds itself out of luck, because at this high level of abstraction the creator losts real control over the algorithms and capabilities.

And that's exactly what happened, and why all of the new concepts and shapes look exactly the same as the other. It needs real effort (maybe even years of learning and practicing) to get out of this *sandbox*, and come up with new, complex and ingenious methods to create something new and exciting.

But whose fault is it? Are the designers all too lazy to move on? Or the software vendors have to make better tools for them? The answer is more surprising than that: the problem is with the current education system.

To solve the above problem, designers need different skills- and mindsets. They have to learn developing softwares. (Even if that only means hacking things together.) Because the tool to solve such problems are already available since day one. The problem is, we do not encourage the students to learn and use them. Furthermore we do not teach them. So it is not that surprising anymore, that they cannot step further on their own. Even to achieve the above described level of knowledge needs tons of extra work in their own free time, because we do not teach those methods either.

Actually there are even more complex problems we are not talking about nowadays, one of the most exciting ones is that how we should use our computers as a standalone artform in our design processes. There are already plenty of great ideas out there, like the one described in Bret Victor's Don't draw dead fish lecture.

And this problem gets bigger and bigger every single year, like a huge spiral, because the amount of knowledge to not only fully understand, but also add constructive criticism to the problem also needs more and more effort.

Prototyping is a key to success

A very important and practical example from designers' points of views is to choose the right process, to produce a product. Maybe the inverse of this problem is even more acute: design a product to be manufactured on specific machines. The challenge could be summarized as: find the reasonable shape, that can only be manufactured by a specific process. And if we are talking about NC machines, and its latest commercial phenomena, the 3D printing technology, we have to design such shapes and textures, which otherwise could not be created by any other machine. That's one of the areas which justifies the design principles of computer generated designs.

But let's move on, and for a second, forget about organic design, or generative, algorithmic form-finding research! If we are talking about everyday devices in our own environments, those are getting more and more complex every day. A single product has more feature nowdays then a car had thirty years before. And we expect designers, to solve these problems, by supporting these features with clean and simple designs, which are also cheap to manufacture and easy to distribute.

We may do so, because we believe, that in a good design form follows function. But how can we create forms, if we do not understand the underlying functions in their full depths?

The only acceptable solution to that problem is to create/design the function as well in harmony with the form and environment of a product. Therefore as designers we have to design and build fully working prototypes, to test them and to learn from them. Mistake by mistake.

To create truly working demos, we have to get familiar, with SBCs, like the most popular Raspberry Pi, or the fully open source hardwares, the BeagleBoards. Not to mention the latest hot topic, the IoT, where the knowledge of microcontrollers, like the Arduino, Tessel or the PyBoard are mandatory.

To build anything meaningful with these, the designer has to be able to *communicate* with them, and that's only possible via programming languages. *(Also some basic knowledge in circuitboards and sensors are also required.)*

And this was just another example, but if we are talking about the next big thing in *design*, then it will be the commercial usages of virtual- and augmented realities. We, as designers of those (semi-)virtual spaces, have to be able to create complete worlds we have never seen before! And we are not only talking about how they look-and-feel, but also how they behave, respond and live with us. Can we really expect from the designers in the traditional sense to accomplish these extremely complex tasks without even knowing what and how exactly the things are happening in these devices and concepts? Or without the knowledge of how to design, build and bring these things to life?

Limitations of the current education

Now, that is a lot to process and master even for the most talented ones. And it is even harder, when there are no specialists around. Fortunately lots of these things can be learnt from the internet (there are plenty of freely available documentations, tutorials and lectures out there) but it can only be done right now, as an extra effort by students who are willing to sacrifice their own free-time to learn all about these things besides their obligatory tasks from their studies. Not to mention, that these methods are way too complex to learn them as they are (from a book or a lecture): one has to go through an amazingly large number of tries and failures. And without a proper support and mentoring, these can discourage even the bravest ones, and makes them give up or stock at a very low level.

Because there are no official classes about these subjects, the students cannot follow *"heroes"*, or their works. A good example, and the fact that they can talk about the same things, at the same level of details is an enormous bonus.

The lack of real research programs, with dedicated and research oriented space and funding makes these things even worse. These kinds of experiments take plenty of time, and needs undivided focus and attention, as well as proper tools, accessories and materials.

Not to mention the lack of motivational tools, because if these knowledges will never be rewarded -- as there are no dedicated people and classes to do so -- it will always be a wishful thinking, that the current education can achieve meaningful goals, even compete with the state-of-the-art methods and as the ultimate goal: challenge the status quo.

The solution

These were the painful lessons we learnt from the last year or so, when we were tried to support and motivate students to move forward and to look beyond the current limitations and start changing things by their own. We almost always hit the walls of being an extra layer on top of the obligatory tasks, or not having enough knowledge to accomplish their ideas.

And this is where the concept of DCL comes in. It needs a dedicated staff and environment. Their job will not only be to teach and operate the facility, but also to experiment with the latest technologies themselves. They had to start their own projects, and if even possible, involve the students at some point.

Speaking of students: DCL has to be a special program for selected students. It is very clear from the previous sections, that this kind of research is not for everyone. (In my opinion, it is not even interesting for most of the students.) The selection can only work, if the students had the chance to learn at least the basics of these knowledges. And only when they have a solid foundation, which they can use and build on top of it, we will be able to select the best ones.

At a small university, like MOME, this can only be achieved if this lab will be faculty-independent and horizontal among the entire educational system. It will work only if it can guarantee, the

long-term research options -- but not as an extra burden besides the existing studies -- but as main one for the selected students.

And this research lab, cannot work locally. It has to have real, living connections between other universities in the country and foreign ones as well. It also has to be able to contribute to the global knowledge -- what it uses intensively -- the free and open culture.

7th of June, 2015