TOPIC 3: SCRIPTING

Burry, M. (2011). Scripting cultures: Architectural design and programming, John Wiley & Sons.

Mathematica
MATLAB
MaxScript
Maya (Maya Embedded Language (MEL) and Python)
Objective-C
Perl
PHP
Processing (Java)
Python
RhinoScript (VB, Grasshopper (VB), Python)
Rhl
VB, VBA

Not surprisingly Maya, Adobe and Rhino appeared in the lists of at least half the group. Among my correspondents there is a cluster of initiates commencing around the year 2002, the time when several packages combined thrilling levels of 3D modelling with associated scripting languages, principally Maya™ (using MEL for scripting), and Rhino 3D™ using Rhino VB. A little more surprising is the fact that over half use languages such as C#, Python and VB, demonstrating real commitment to the task. *Processing* (shareware), written by Casey Reas (one of my correspondents) and Ben Fry, is used by half my sample. The power and simplicity of Processing point to the phenomenon of an emerging generation of scripters who wish to focus their intelligence on sorting out the logic of their design approach rather than on hitting obstacles with obscure coding syntax. Processing is a language designed by a designer to help other designers leapfrog over the lack of ease of use that characterises many programming languages.

To script or to brief others to script

I was surprised by the admission of several correspondents that they are no longer in a position to take up any new programming environments as their careers have moved on to leadership roles with associated time constraints. Others will not relinquish their direct pivotal role in design scripting at the highest level on the basis that as designers they need to be pulling, not pushing.

Many of those consulted have had access to expertise that can vary from straightforward assistance to (as in my case) input to coding at a level I

would never have reached by myself, certainly within the timeframes of the projects concerned.

Challenge or breeze

With only a single exception everyone consulted sees programming as a hard-won skill, not acquired with ease. A few ventured that their teaching experience reveals that some students are more naturally able to assimilate the rather unusual aspects of code writing (compared with all other design activities) than others. Some suggest that on the one hand scripting is a vital opportunity but on the other, the required logical approach is challenging for designers more familiar with freer ways of thinking. There is evidence of people who, while they 'get' scripting, and despite scripting very ably, still struggle with it. With unusual candour some of my correspondents admitted to struggling themselves, all the more astonishing when their top-drawer output is compared with a self-assessed lack of facility – another instance of hope for the reader who might suspect that they too are a potential struggler. Most replied that it gets easier as they progress and, as with any high-level skill, constant practice is a necessity. The comment I enjoyed the most in this regard was that scripting is easy until you get to the first bug.

Design productivity and design exploration

I was also curious to learn the motivation of the designers I approached – what attracted them to scripting? The responses were satisfyingly varied:

- reaching beyond analogue processes;
- capturing material logic and computing performance;
- being playful;
- exploiting generative processes;
- seeking deeper access to the imagination;
- engaging with complexity;
- inducing rapid iteration and variation;
- grappling with the performative;
- toying with the unexpected and delving into the unknown;
- being forced to be explicit;
- discovering novelty;

- localising intelligence;
- investigating self-organisation principles;
- studying phenomena;

… and of course going for good old task automation.

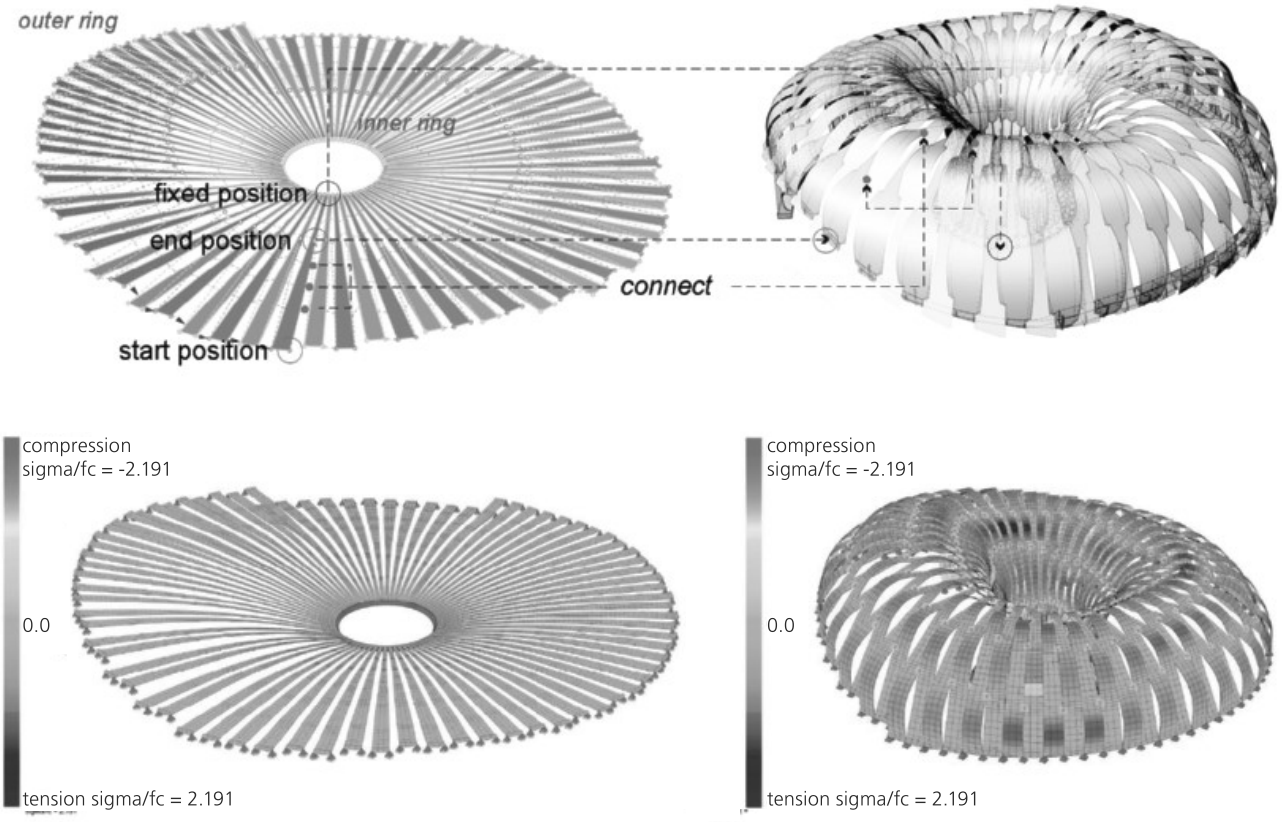
Hardly two responses were the same, which is a circumstance I explore in a little more depth in the book's conclusion.

I received a similarly wide and perceptive range of responses to my enquiry about potential dangers in scripting such as contributing to 'the death of the pencil sketch', 'enfranchising the amateur', or design becoming 'automated'. In some it provoked a response bordering on irritation – as if this were an absurd proposition – while others went to the other extreme, positively welcoming an end to the supremacy of design skills being framed entirely around mind–hand–eye coordination. Several made comparisons with the arrival of cheap high-definition still and movie cameras, which they noted had not necessarily reduced the quality of top photographers and filmmakers: talent always shines through. This is an optimistic view perhaps, as one considers the implications of the architectural equivalent: amateur house designers given tools to create their own home; can we really compare an album of poor photos with a landscape blighted by properties built from poorly understood design principles? Is this even a scripting consideration?

There is at least a niggle that a designer who has appropriated the use of someone else's script to create 'interesting effects' might end up with the credit for something for which they are not the complete author. Again, this is the dilemma of tool or process versus outcome. When scripting there is a further dilemma that comes from the typical need to begin encoding a design at the outset compared with new ideas emerging through intuition from the undirected sketch, as it is hand drawn or modelled along the way. But in scripting a design, a logic has to be investigated up front, which for many can be seen as an opportunity, not a hindrance. Any argument that scripting leads to standardisation can be countered by the fact that it affords highly wayward and idiosyncratic designers the opportunity to innovate in ways otherwise not possible, which segues into my next line of enquiry: new design avenues.

Essential scripting and its value

When I asked my correspondents what they could design through scripting that they could not reasonably produce otherwise, and with what added value, I received a highly spirited response with as many unique suggestions as respondents. Firstly, the ability to work with large data sets, proceeding in many directions simultaneously, and working beyond our perceptual capacity were prominent responses. While scripting might veer towards the complexification of otherwise quite simple things as well as deal with still more, several pointed out that scripting can do the exact opposite, and be used to look for simplification. Others enjoy the opportunity to capture specific know-how, encoding it with tacit knowledge rendering it more declared and shared. A proportion of my correspondents also pointed to the link between scripting and fabrication, which is something that I deal with in detail in chapter 8.



Achim Menges and Jan Knippers, ICD, ITKE research pavilion, Stuttgart, 2010.

How stuck would scripters be without the opportunity to script elicited replies ranging from not being able to do any of the things that particular individuals or practices currently undertake to do, to only being inconvenienced, forced to spend more time than would otherwise be necessary, and possibly being forced to work with a reduced set of inputs.

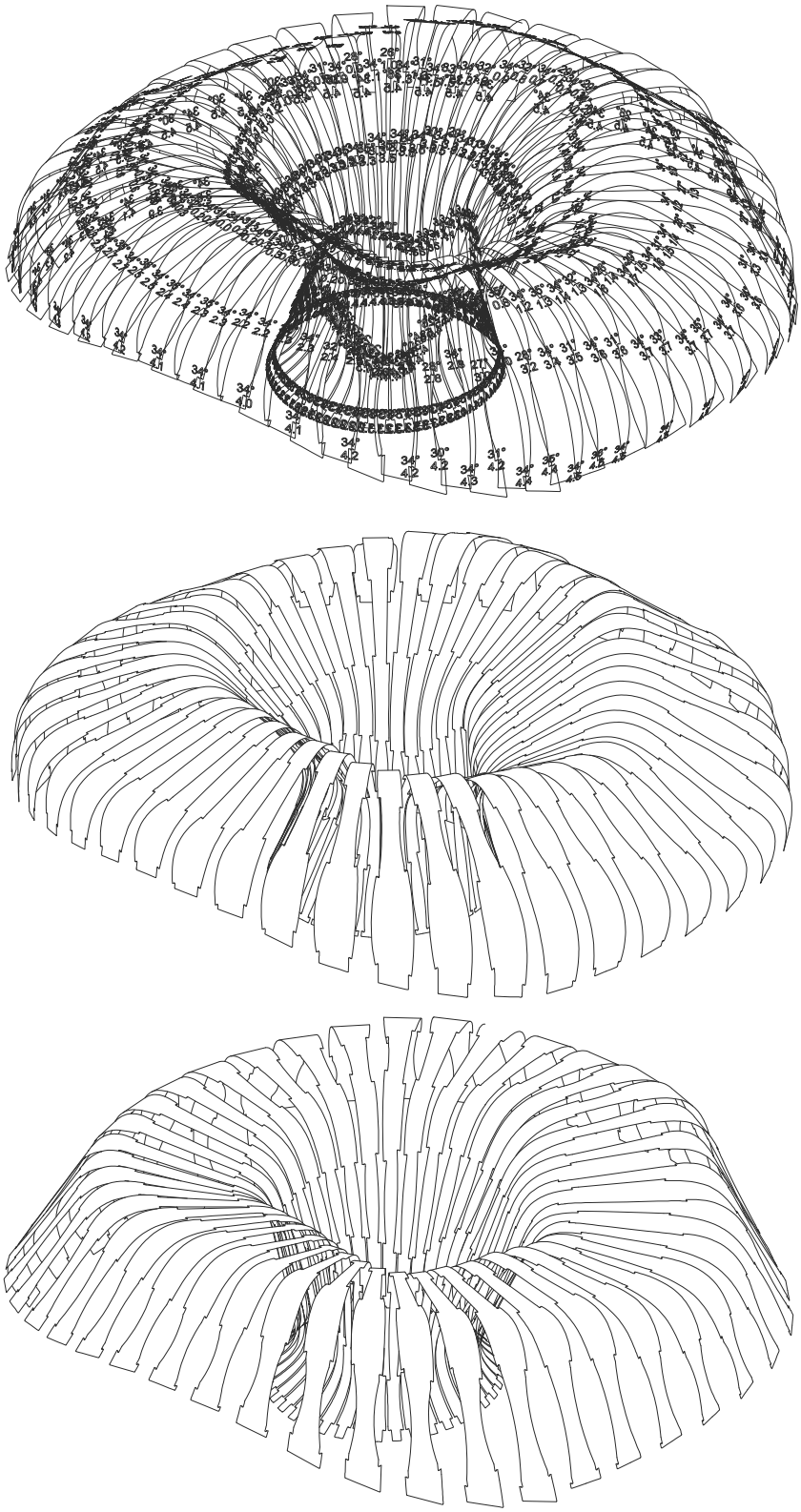
Scripting education

In this chapter I have covered the origins of my own interest in scripting, and I have opened up the subject a little by anonymously reporting the opinions of a group of experts who have responded to my request for additional insights.

My motivation for writing this book includes the proposition that one generation on, scripting seems to be here to stay. So what does this mean for the education of architects? I conclude this chapter by incorporating the views of my correspondents listed earlier to enrich what ought to be a very vigorous debate in every school of architecture and all practices still relying on traditional ways of working in the face of a wider uptake of scripting.

The need for prior knowledge

I am impressed by the way that architects' professional organisations in many countries have worked hard to ensure that a school that is highly experimental and culturally focused is professionally accredited on the same basis as a school with a strong technical focus, for example. Correspondingly, entry requirements seem to vary considerably. I have worked in schools where maths proficiency at senior school level was a core requirement and others where no such condition of entry is made. Given that almost all those I have consulted see scripting skills as being hard won, should the schools that include scripting at an elective if not fundamental level seek some computational skilling as a prerequisite to coding? If this comes across as an unusually draconian proposition, readers who have tried to involve in their scripting studio a student with no maths beyond early senior school basics in a class otherwise full of those with a good level of maths, will appreciate the dilemma that prompts this question. For anyone nervous that compulsory scripting skills for pre-university initiates is a secret agenda within scripting cultures, the selected responses that follow will help allay any such fears, while still provoking some discussion all the same.



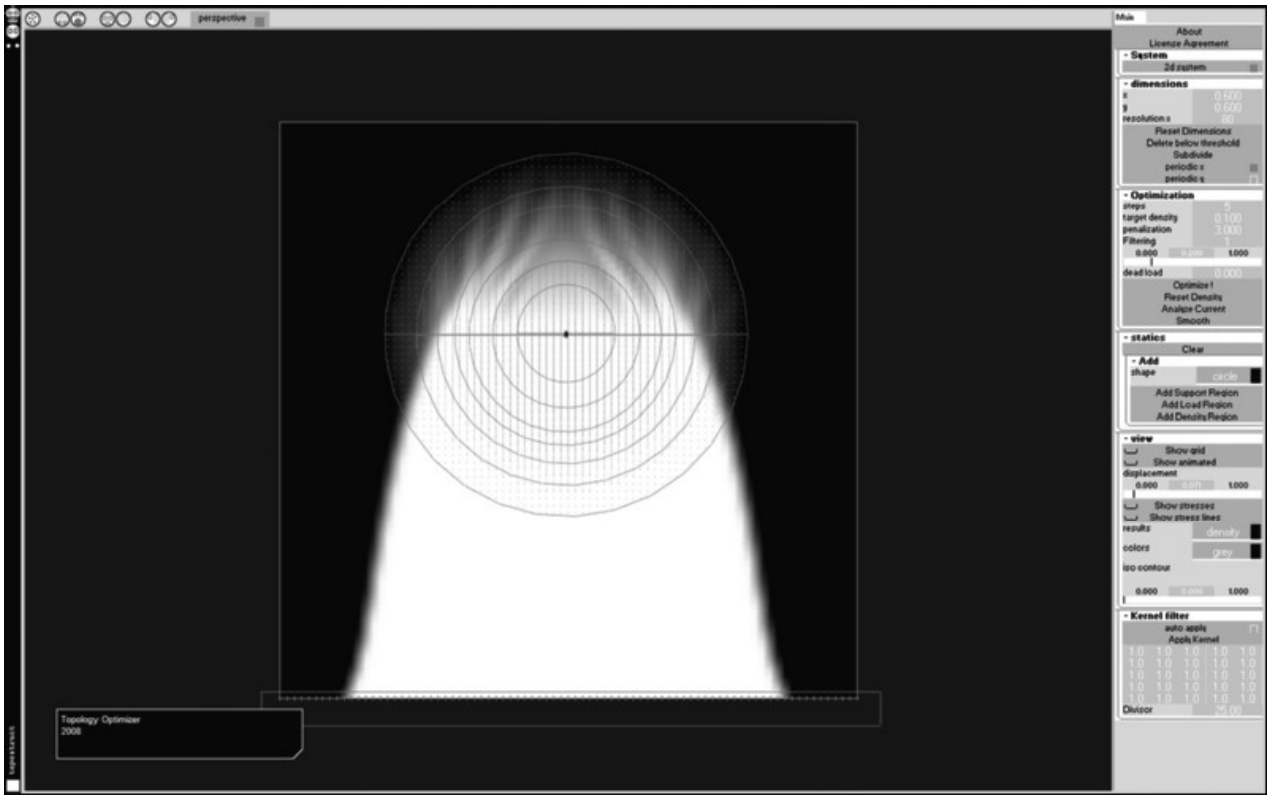
Achim Menges and Jan Knippers, ICD, ITKE research pavilion, Stuttgart, 2010.

The question does open up a broader educational issue, for instance the assertion that scripting might contribute usefully to what is taught in many disciplines well before college.

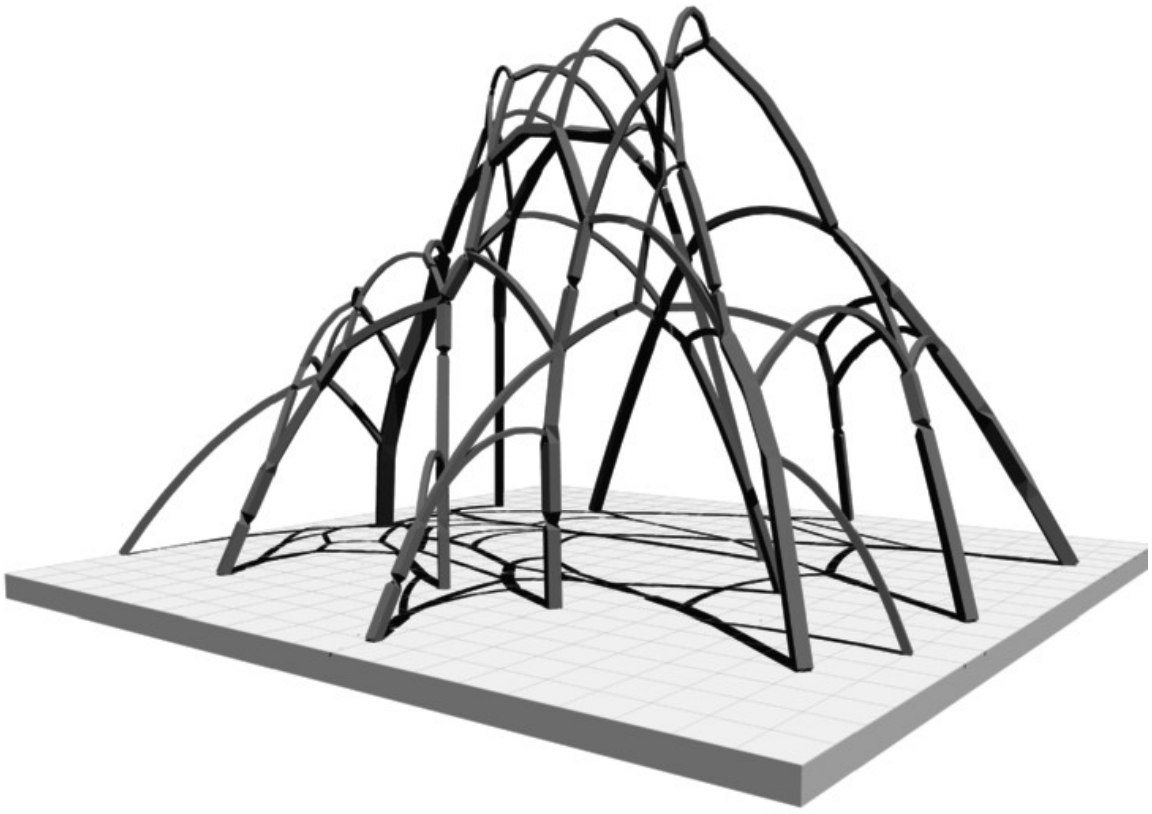
> *Scripting should be taught in high-schools (will elementary schools follow) as general culture: scripting for architects & designers, scripting for mathematicians & biologists, scripting for dancers & musicians, scripting for poets & moviemakers & dramatists.* (Vito Acconci)

There are also assertions that procedural literacy has an increasingly vital role to play:

> *Yes, absolutely, I think the education system should start teaching procedural literacy from the beginning. Students should then build on these skills and focus them within the context of architecture.* (Casey Reas)



Panagiotis Michalatos & Sawako Kaijima, AKT Architects, Topology optimiser, London, 2010.



Axel Kilian, Vaults, particle spring library Simon Greenwold, MIT, 2004.

This level of affirmation is not universal with a call for sufficient scope to accommodate the unexpected perspectives that innocence often brings to the mix:

> *It is useful but not necessary at the moment. People with no skill or interest in digital media may bring a different and often needed perspective or experience.* (Panagiotis Michalatos & Sawako Kaijima, AKT Architects)

Crucially, many argue the case for prior training in thinking procedurally rather than for arriving with prior coding ability:

> *What is far more important than the mechanics of scripting is for students to be able to think algorithmically. One question might be: Should this be taught at high school or at University? We then might consider the subsequent issues as to whether algorithmic thought should be taught abstractly or as applied to a specific subject (of interest to the student)? Therefore linking the algorithmic thinking to the subject of architecture might make it more attractive to some students who might not respond to the abstract form.* (Robert Aish)

There are also those who are happy to work with all comers; note the bravura registered with the following two comments:

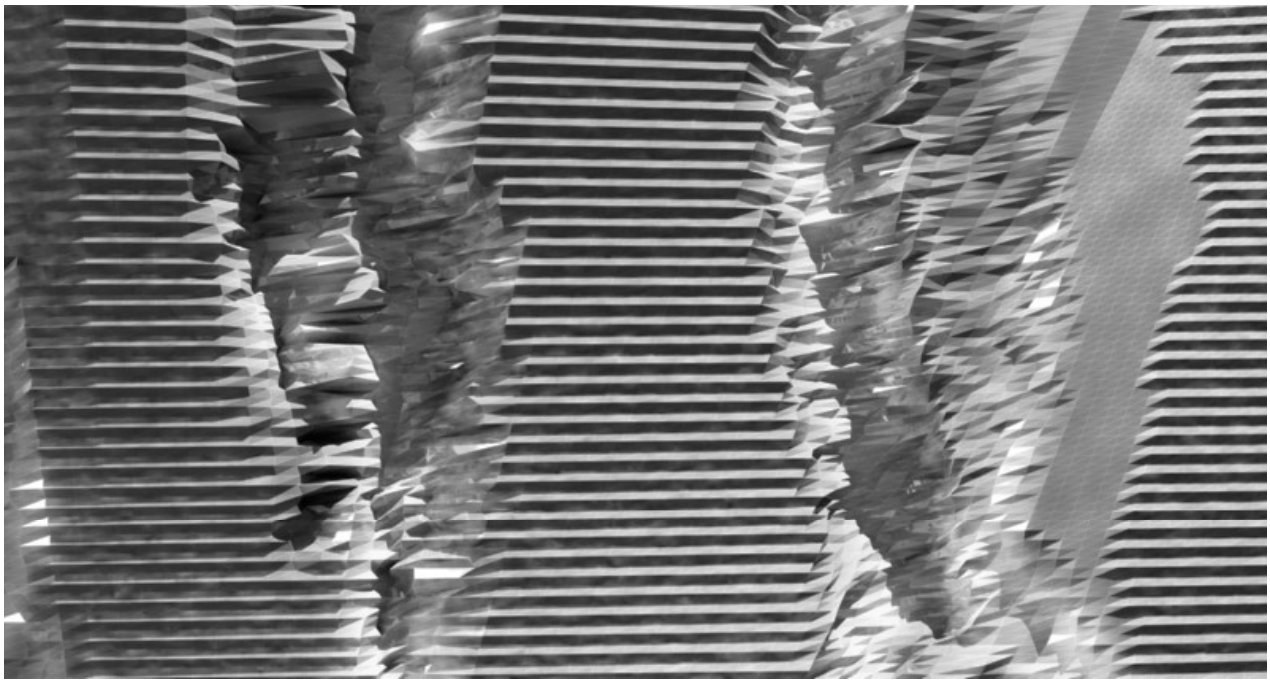
> *Not necessarily. They can 'mutate' very fast. It is often better to learn scripting and abstraction layers directly on design problems.* (Alisa Andrasek)

> *No, it's OK for them to learn it there – it's part of a culture. And those that have familiarity through another education have to retrain themselves anyhow.* (Peter Macapia, labDORA)

There is potential for a debate here: the hacker and masher getting by versus the highly skilled writer of elegant code working with the benefit of thorough learning. As useful as the latter context might be for skilled grounding, it is not deemed essential by the community, nor is it likely to become the norm.

Scripting as part of an architectural education

Although there was ambivalence about prior learning, almost all my correspondents are quite emphatic about the need to integrate scripting into the curriculum, although 'learning by doing' through direct application in



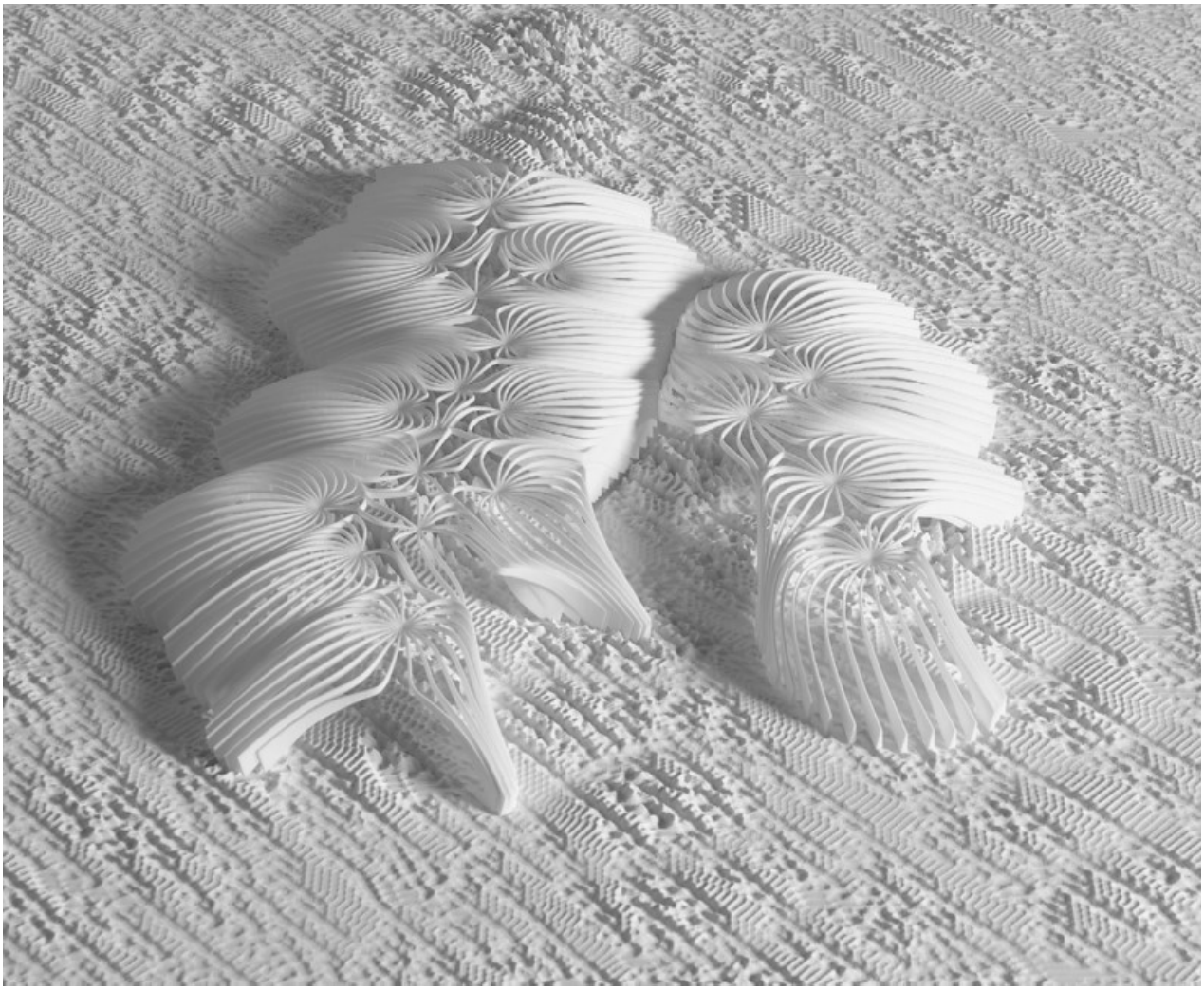
Biothing, Fissure – Agent Wall, London, 2010.

the studio is seen as an appropriate environment by many. As shown in the selected extracts, some proponents of the teaching of scripting in college are quite emphatic about this.

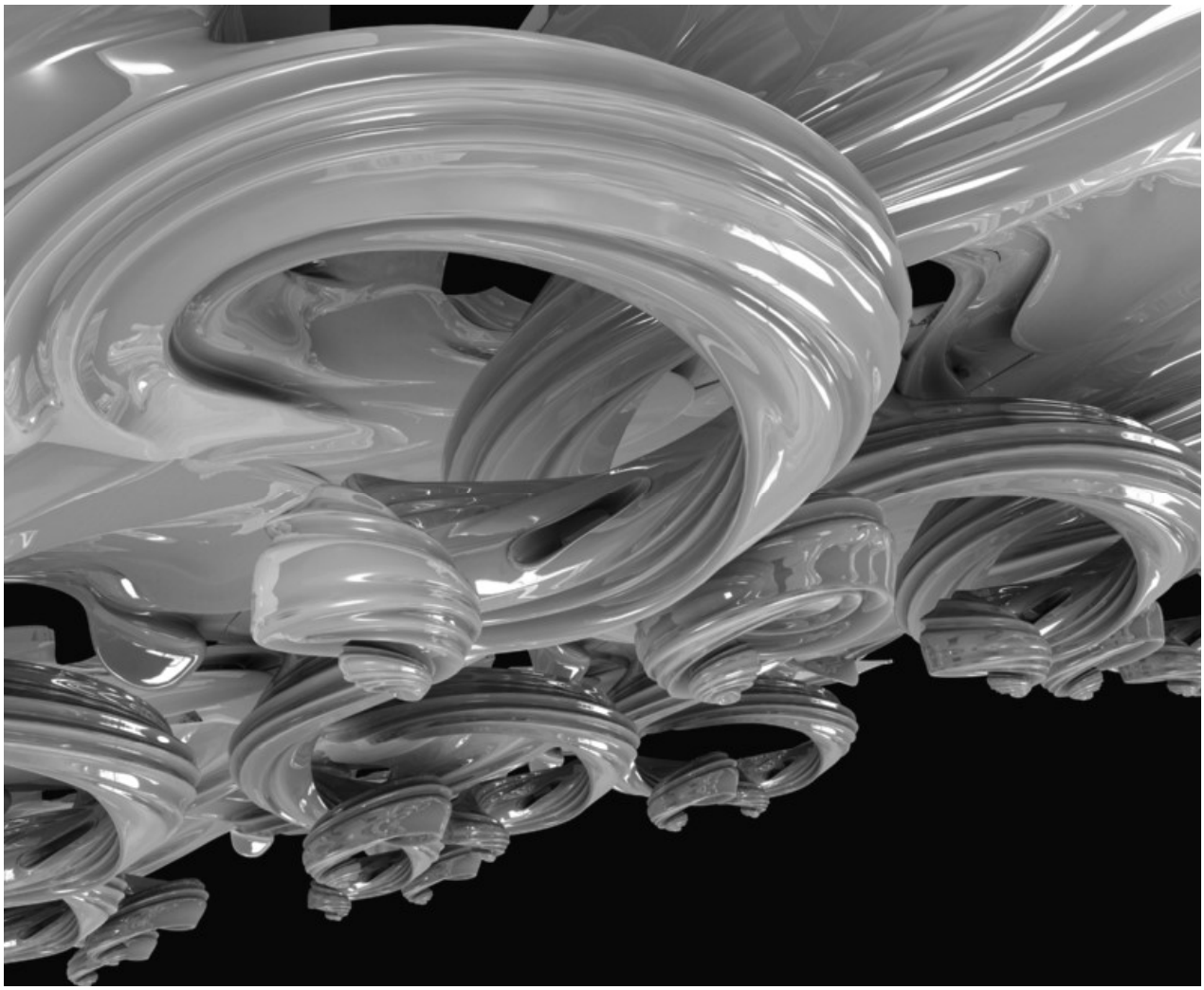
> *Architects are ultimately choreographers of systems, and the benefits of teaching programming in an architectural context are manifold. If architecture wants to survive as a discipline, it needs to engage the culture of innovation and computing.* (Mark Collins & Toru Hasegawa, Proxy)

Scripting can be seen as offering an alternative view of creativity:

> *An algorithmic understanding of creativity – that the act is not a flash in the dark, a blessing from heaven but the result of hard rigorous thought which*



Biothing, Seroussi – Mesonic Fabrics, London, 2009.



Evan Douglis, Helioscope, New York, 2008.

> *can in many parts be represented algorithmically – that is the important step. Right now, scripting is a very useful paradigm but it is not exclusively so.*
> (Tom Kvan)

And a call for scripting education to be subsumed in a wider intellectual framework:

> *To the extent that it is framed within the intellectual development of language, writing, mathematics, etc. and understood for what it is, not mythologized.*
> (Mark Goulthorpe)

These are still strange times for architectural pedagogues settling down from the previous tensions between analogue as opposed to digital design

Evan Douglis, Helioscope, New York, 2008.

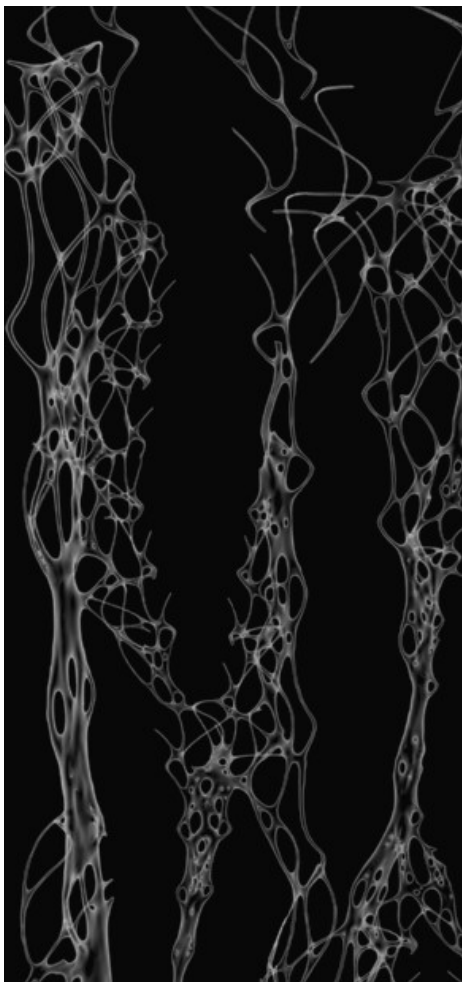
practice, who now face another unscheduled cultural shift. While we are moving well into an era of digital design acceptance, we nevertheless operate within a legacy that includes many educators who have no idea of how to do anything beyond the basics with their computers. If we obsess about the need to teach coding skills we will repeat the errors of the 1990s when CAD equalled drafting. Those who want to script need to be taught by designer scripters and not by 'computer people'. More crucial still is the need to focus

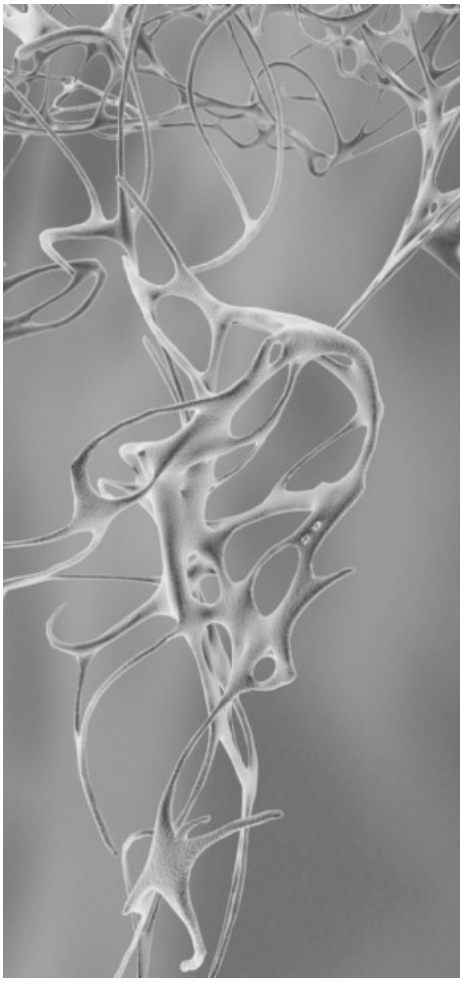
beyond ‘scripting’ to the meta topic: an appropriate approach to learning about the emerging systems in which scripts operate; culturally, and as an emerging theory.

Scripting critique

Given the wealth of opportunities that scripting offers to shift design practice in new directions, can we anticipate significant shifts in architectural culture and its critique, or will scripting simply be subsumed once the novelty has gone?

Since the cultures of scripting are evolving they are not yet fully formed core design constructs, hence my investigation into cultures rather than probing into ‘scripting’ as if it were some kind of *Zeitgeist* movement. Two





Kokkugia, Swarm Matter, New York, 2010.

Kokkugia, Micro, New York, 2010.