ALGODE TOKYO 2011

Proceedings of the International Symposium on Algorithmic Design for Architecture and Urban Design,
ALGODE TOKYO 2011
March 14-16, 2011, Tokyo, Japan

Fig. 1: Rendering from the "Digital Design Methods" website.

Why MEL?

Ass.Prof. Dipl.Ing. Richard DANK

Institute of Architecture and Media
Graz University of Technology
dank@tugraz.at

Abstract

In the contemporary architecture office and other cutting-edge cultural industries the practical demand for creative programming is plain obvious. On the one hand the automation of recurring routines requires those skills. And on the other it is the promise to gain de facto self-determined access to information processing, advanced design and building methods that generates demand. In the year 2011 we may accept this as an undisputed fact, at least within the digital community.
But apart from that this paper argues that an algorithmic approach offers students as well as teachers new strategies to explore their ideas, develop their conceptions and enhance their modes of thought. And this amplification of reasoning is the focus of our interest.

Within the paragraphs to come I will call several witnesses to the stand to demonstrate why we think scripting - and I am talking about alphanumeric code and not graphical editors, tools or plugins - should be an essential, mandatory part of any architects' education in this new millennium. Moreover I will try to support this hypothesis by documenting the experience we

have gathered working on this matter for over a decade now. And finally I will attempt to illustrate what might be an effective, juicy and playful process to take the first few steps into parametric, algorithmic design.

As a matter of fact we will transmute data into "nodes with attributes that are connected" via Maya Embedded Language...
Case study: Scripting high-rise towers with architecture sophomores.

Keywords

Project and Practical Application of Algorithmic Design
Computational theories applied to design
Generative Geometries and Parametric Design

1 Introduction

In the foreword to Maeda@Media Nicholas Negroponte (2000) [1] recalls that during his architecture studies the most profound thing he learned is to ask good questions and not to give proper answers. Moreover it occurred to him that at the end of the day improving the design tools - or rather upgrading the design strategies turns out to be far more rewarding than designing a few good buildings.
But what may be considered an appropriate, promising approach in a Post-Industrial Society, where according to Daniel Bell (1976) [2] theoretical knowledge is the center axis sophisticated technologies revolve around? And how should it be taught on an academic, collegiate level?

2 Digital vs. Computational

In 1967 after joining the faculty of MIT, when the Digital Revolution had not really come into sight yet, Negroponte founded the Architecture Machine Group and two decades later the Media Lab. And this was exactly that environment where John Maeda (2001, 19-20) [3] published Design by Numbers. Back then he was not happy that "most of the parties involved do not realize that computers, as they are used today, have nothing to do with design skill, or design education for that matter". Subsequently he craved out that of course "mechanical skills have taken secondary importance to the skills required to use complex software tools. But what is the nature of these digital skills, and more importantly, are they really of any significance?"
Even in 2011 it seems to be common currency that a recognized digital designer is somebody who masters the most advanced software packages beyond compare. Naturally the computer industry takes the same line. But Maeda (2001, 20) [3] argues that these skills "are nothing more than knowledge, and that we implicitly glorify rote memorization" deriving from endless tutorials, how-to manuals and books for dummies.
But "the true skill of a digital designer is the practiced art of computer programming, or computation". Well, I could scarcely put it any better.

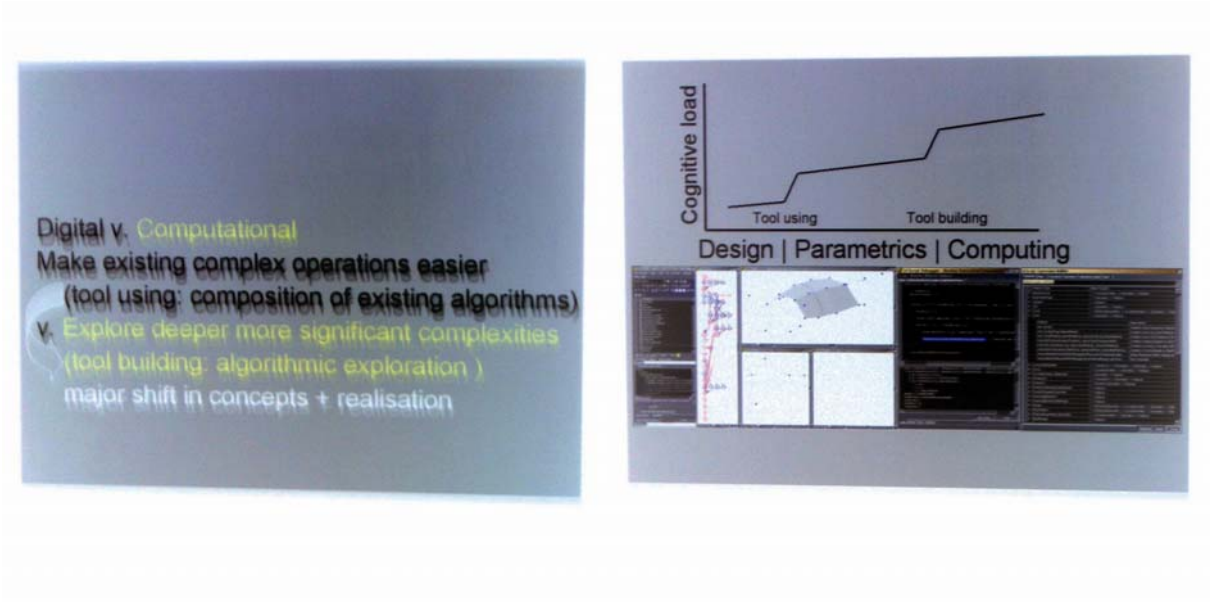

Fig. 2: Photographs taken from Robert Aish' presentation at the Game Set and Match II Conference.

But Robert Aish, I borrowed this chapter's title from, expresses it even more emphatically. In his initial presentation at an evening discussion panel within the Game Set and Match II Conference he pointed out that: The "Digital […] makes existing complex operations easier". The "Computational […] explores deeper more significant complexities". Bernhard Cache, John Frazer and Mark Goulthorpe plausibly supported this position on the same occasion (Dank 2006, 83-85) [4].
Finally it boils down to this. One can use this undeniably handy apparatus to draft comparatively straightforward and obtain a certain result reproducing one's preconceived brainchild. Or you could use the Turing Machine - and a contemporary computer is fundamentally nothing more than this theoretical device - to combine raw scripting skills "with your enhanced creativity, and the result will be unique explorations of the digital medium that fully exploit its true character" (Maeda 2001, 21) [3].
Keep in mind that we obviously support the latter. And let's move on to the next major question of this paper.

3 Algorithmic thinking

When I say 'we' I mean Andreas Gruber and me - previously my colleague at the Institute of Artistic Forms and now at the Institute of Architecture and Media headed by Urs Hirschberg. Andreas is working algorithmically since the early 80s and teaching in the vicinity of this subject since the mid 90s. We teamed up just before the turn of the millennium and have been collaborating academically and beyond since then.

In the previous chapter I pointed out that programming is the congenial way to explore the possibilities of the binary algorithmic machine. But apart from this fact what has crystallized over all these years is that scripting sharpens the mind even for non-digital tasks. Though this thesis ought to be issue of another paper and might in fact fill books, I will try to outline this crucial aspect within a few sentences.

In one way or the other architectural design is concerned with abstraction - no matter how (non-digital) you work. Following Scott McCloud (1993) [5] the transportation of information ranges between reception - like the understanding of an image - and perception - like reading words describing the scene, where you obviously need specialized knowledge (about the characters, the syntax, the language, et cetera) to decode the content.
In architecture even a physical model can only be a close cast of the completed building. And it merely suggests the three-dimensional setting. Moreover it takes a scholar to get the message from a floor plan or a cross section with all their icons and symbols. And the rough freehand sketch is sometimes even hard to apprehend for the one who conducted the pencil. But all these methods and tools - in good approximation of reality or recklessly nonrepresentational - help ourselves understand our own projects and refine our ideas while designing. Or as some may put it: The brain learns from the hand – and vice versa.

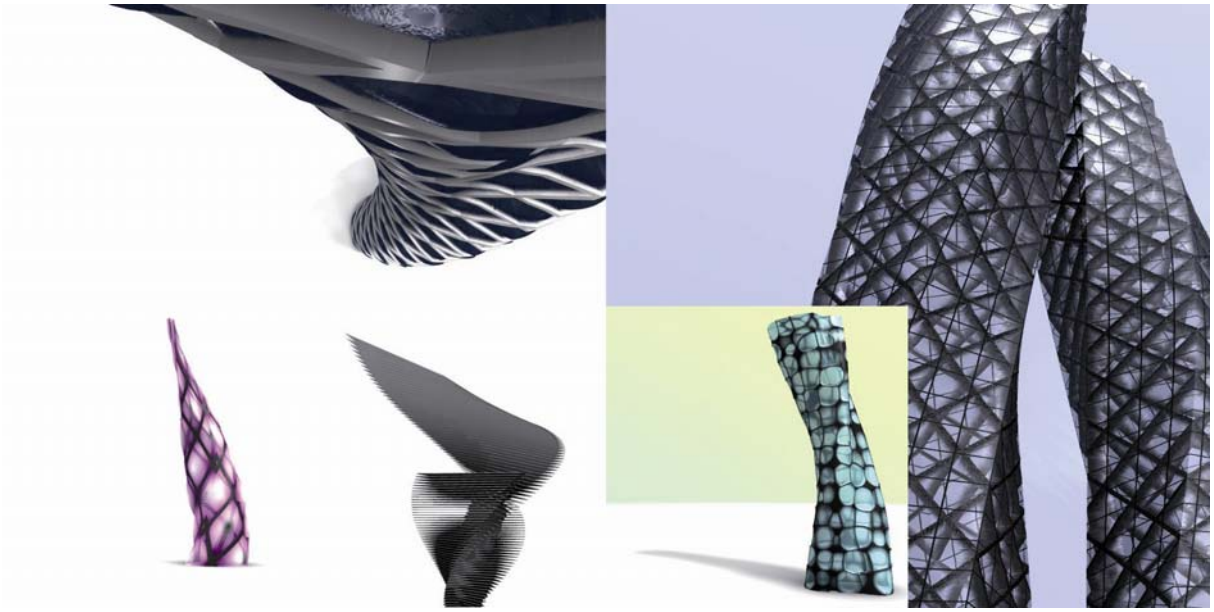

Fig. 3: Assignment A "Stack and Staple" (Maya) and B "rise high" (Mat+Map). Experiments texturing the surfaces generated by the toolbox provided.

Scripting - converting into code, into commands, variables and functions - can be assumed to be the most abstract coevally most precise way of describing something. You need to know exactly what your ideas really consist of at root to transcript it into a programming language.

However the way to a fully working digital sketch is never linear. It's back and forth, destilling your strategies in order to improve the result. Thus the "cognitive load" can be much higher and of substantial meaning - see Figure 2.
But there is even more to it. On the one hand you cannot just scribble a line. You need to set distinct values. On the other you can genuinely have it generated at random. And instead of hard-coding you can compile parameters and cross-refer them. Basically you encode all your approaches and concepts - not ready-made, concluded bits and pieces - into a formal language and watch what develops.

The point is that scripting compels you to ponder thoroughly. It makes you distinguish between the hard and the soft facts - and their relationships. It forces you to precisely formalize your general set-up and all your framework conditions. It opens up a large field of possibilities and does not just accelerate customary approaches. Thereby it's irrelevant to write uniform, standardized, high-profile scripts from the first. Beautiful Code comes out of practice. And if creativity is solely situated in the right cerebral hemisphere, it will also set your left mathematic-logical part on fire. The enhanced correlation of mind and brain then ignites all your neurons and lightens your design.

To sum up that chapter: I claim that people who had first hand experience with typing their own Creative Code subsequently have higher control of their proceedings and finally better understanding of their results even when planning on a sketch roll or out of their scrapbook.

4 Now seriously, why MEL?

Contemporary design methods - the process of originating and developing - could be described as putting entities into relationship or interrelate ideas within a certain system or correlate - if you will. The pioneer Friedrich Kiesler (1949, n.p.) [6] verbalized it long after the Raumstadt like this: "Each element in a construction or in a city […] is conceived not as the exclusive expression of a single function but as a nucleus of possibilities which will be developed through coordination with other elements. Either this correalation can be based on physical conditions or else on environmental conditions or else even on the very essence of the actual element."
The rediscovery of Kiesler's theories in the 90s does not seem to have happened by accident. Omnipresent advances in architectural experiments with the computer rendered his vision of the Endless House possible. "The complexity of the form and content" - "brimming of information" (Bogner 1997, 21) [7] - could now be implemented. And even Maya's catchphrase "nodes with attributes that are connected" could well be of his origin.

But of course this is not the reason why we finally turned to MEL for teaching the basics of parametric, algorithmic design. As any language Maya's Embedded Language has its specific peculiarities - ironically called Melisms -, which could be annoying at times. But although it is descended from shell scripting, you could rather quickly manipulate the data structures and develop your own functions, even when the online help doesn't suggest so. (Autodesk 2010, MEL for programmers) [8]

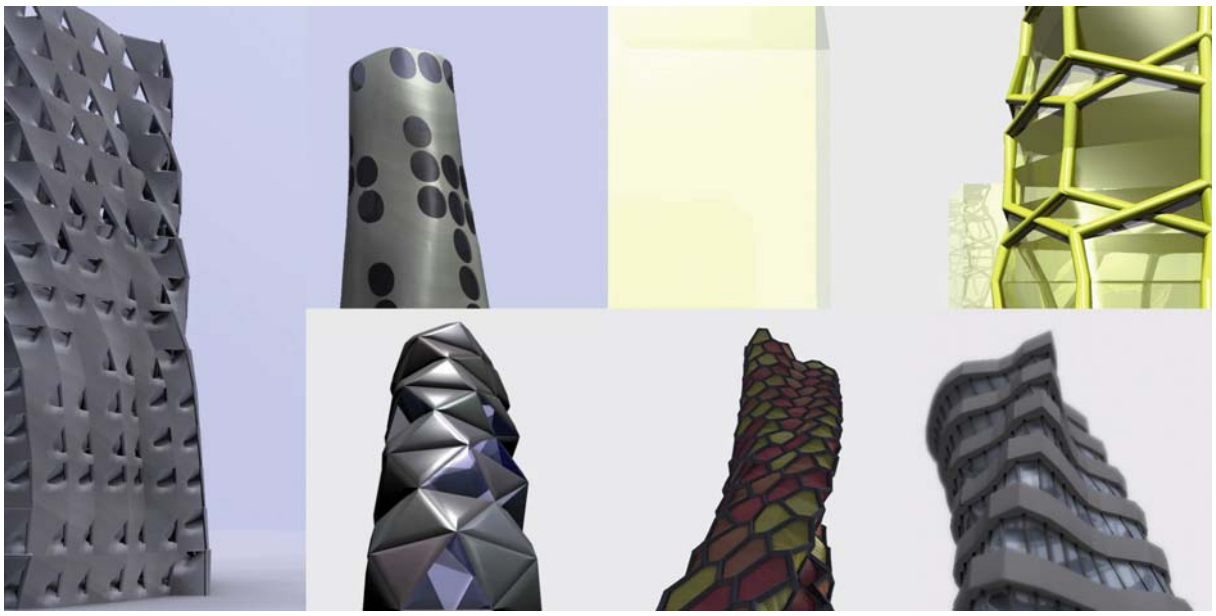

Fig. 4: Examples from assignment C "LIStProcessing" (MEL).
The potential of reading, writing and manipulating array data.

I don't want to go too much into detail, but most important for the sophisticated user: "MEL is a scripting language at the heart of Maya." (Autodesk 2010, MEL Overview) [8] This means you can control the complete software package and even access advanced features, as opposed to many parametric graphical editors, which have their natural limits. And for the rookie programmers, we are talking about here, there is the Script Editor. It returns any command and/or result in the top pane - even if you call them by clicking a regular interface button. In fact all interactions execute MEL-commands or small routines.

So you almost literally peep under Maya's bonnet. Plus you get profound knowledge of any object component, any data connection, any modeling method, any render algorithm and so on.
Above all the modus operandi and the syntax are rather universal, so one can easily switch to many other scripting and general-purpose languages. MEL-dexterity scales big time! Whereas Grasshopper for example only applies to Grasshopper. Potentially also to PD and MaxMSP, but that's about it.
And finally it is apparent that Maya is fitting architects' requirements perfectly as it essentially is a 3D modeling, animation, and simulation application.

Fig. 5: Schedule of the winter term 2008/09. Left column: The arrangement in 5 blocks with the basic focus. Right column: The Name of each seminar with its content.

STARTUP	
Plenum	Introduction to the issues of the semester for all groups
Block A Maya 1 seminar	**"Morph"** MayaGUI (Prefs, Menu, Panels, Hotbox), Curves (QWERTY, ChannelEditor, MarkingMenu, Components), Animation (Blend), MEL (ScriptEditor, "nTowers.mel"), VectorRenderer
AsA Assignment	**"Stack+Staple"** (For examples of the students' output see Figure 3.)
Block B Mat+Map 2 seminars	**"Shading Network"** Polygons (Components, ContructionHistory, Outliner), Materials (Hypershade, ConnectionEditor, AttributeEditor, ShadingNodes) **"Lights. Camera. Action."** Materials (ShadingNodes), Lighting (DirectionalLight, DepthMapShadows, RaytraceShadows, domeGI), Cameras (withAim, DepthOfField), MEL ("rbTowers.mel"), SoftwareRenderer
AsB Assignment	**"rise high"** (For examples of the students' output see Figure 3.)
Block C Mel 3 seminars	**"lOOP"** Introduction into MEL, SciptingLogic, Variables (Type, Value), Loops (for), Conditions (if, else, Operators), MEL ("_library.mel") **"aRRay"** Variables (global, local), Eval, Arrays ([ ], size), Procedures (Arguments, Returning) **"MELism"** MEL SpecialCases, (noteable Banana Skins, Backticks, whatIs), xform, short Routines (Calculations, Angles, Coordinates, Lists, Curves, Surfaces), SampleScripts for Assignment C
AsC Assignment	**"LIStProcessing"** (For examples of the students' output see Figure 4.)
Block D Mel 3 seminars	**"Hands on Lists"** MEL (Repetition of Block C's contents, Developing own Routines, Access to Lists, "Kugerl_und_so"), SampleScripts **"Interactive Panels I"** MEL (Repetition of Block C's contents, AttributeConnection, Expressions **"Interactive Panels II"** theoretical and practical PreliminaryWork for Assignment D, 'LiveScripting', SampleScripts for Assignment D

AsD Assignment	**"Parametric Panel"** (For examples of the students' output see Figure 6.)
Block E Acrobat 3D 3 seminars	**"OUT layers"** Illustrator (Introduction, Authoring, LayerOrganisation, Interpaly with CAD, PDF-Export), Acrobat3D (FormElements, Hyperlinks, Visibilities, Controls) **"ANI mate"** Maya (Grouping of Objects, DataExport), Acrobat3D (3D-DataImport, Configuration, Perspectives, Sections, Visibilities, JavaScript) **"Hyper-Narrative"** Acrobat3D (Repetition of the contents especially important for the Final Presentation), Maya (SpecialCases), Questions?
AsFin Assignment	**"Hyper-Portfolio"** (For examples of the students' output see Figure 7.)
FINAL Plenum	Final presentation and critique

5 TUERME.n

To top things off I picked out one of the compulsory courses (worth 3 ECTS) we gave recently to show how these themes could be approached in practice. The TU Graz is one of very few architecture schools worldwide that oblige every single student to learn at least the basics of programming. And these "Digital Design Methods" (== name of the course) I am referring to took place in the winter term 2008/09 with Christian Freissling, Andreas Gruber, Urs Hirschberg, me and more than 370 (!) students clustering to experiment with generated high-rise constructions. "TUERME.n" is therefore a pun in German for towering up and furthermore indicating n (a whole lot of) towers.

That particular semester the undergraduates were to erect virtual edifices and clad them with panels. The additional aim was, that the subjacent skeleton consisting of vertices and curves and the individual facades should be interchangeable. Hence online Creative Collaboration (Schmitt 2001) [9] was another demand.
We split the course into five blocks or four parts with three seminars each. (See Figure 5 for the schedule.) To commence things appropriately we had a Startup plenum and ultimately a final presentation and critique of the resulting projects. After each block the students had to upload an assignment to illustrate their progress.

As every year we started out with an appetizer - some routines with a front end we prepared to give a notion of the possibilities and the things to come (Block A, 1 Seminar). Then we introduced Maya's backbone (Block B, 2 Seminars) - the attribute-node-coherence - using the example of the shading network and rendering capabilities. (See Figure 3.)

Afterwards we took them down to earth again, only to teach them how to fly with MEL on their own - if you'll pardon this flowery expression. We believe beginners absolutely need to set about scripting at the basics - with variables and arrays, with loops and conditions, with procedures and expressions (Block C, 3 Seminars). Commencing with tiny snippets, which are easy to understand, reproduce and adapt, we gradually move on to complex constellations (Block D, 3 Seminars). In the end - right before Christmas - they ought to be able to manipulate and construct their own data and code - in that case build algorithmic towers. (See Figures 4 and 6.)
Finally the tool of our choice for a proper presentation is usually Acrobat 3D, offering several very interesting options to interact (Block E, 3 Seminars). (See Figure 7.)

We still consider the outcome of the course to be very successful and inspiring. But you might want to catch a glimpse at http://iam.tugraz.at/dm1/w08/ to get an impression of the large variety of the different concepts and make up your own mind.
After all the promising performance was another testimony that those strategies are of major importance for the future development of the chair. Subsequent elective subjects and above all our Design Studios extremely benefit from the fundaments laid in the "Digital Design Methods".

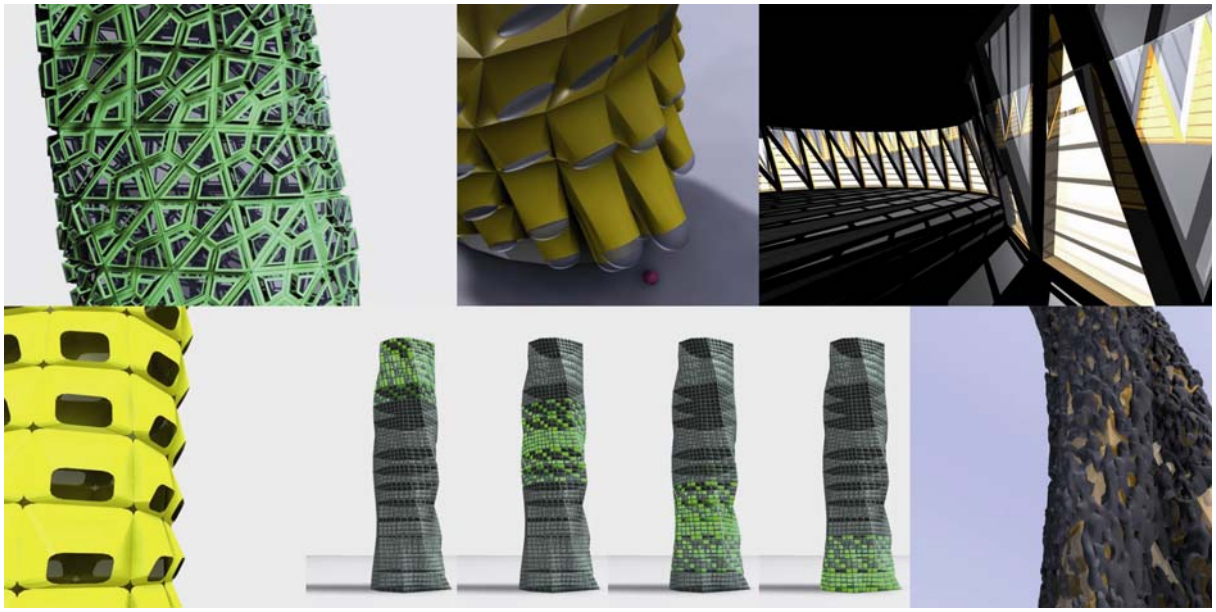

Fig. 6: Assignment D "Parametric Panel" (MEL).
Algorithmically designed cladding. Certain facades are even responsive.

6 Conclusion

Bernhard Cache locates the origin of parametric design in Vitruvius' De Architectura. He claims in the recently published Graz Architecture Magazine that the "author of the only treatise on architecture that has been handed down to us from antiquity" "determined the different components of a building by means of numerical relations" (Cache 2010, 52). [10] More than 2000 years later the ongoing development of the computer hard- and software is providing today's designers with continually growing possibilities to effectively model these algorithmic approaches.

But does that mean that it makes sense to parameterize even the smallest task? Of course it doesn't! But future architects definitely must stand up to sophisticated problems. Even if they ultimately do not program every single line on their own - as they do not compute the structural calculations. But they need to assimilate a deep, fundamental comprehension of this particular matter. And that can only be grasped by getting one's hands dirty - with raw code!

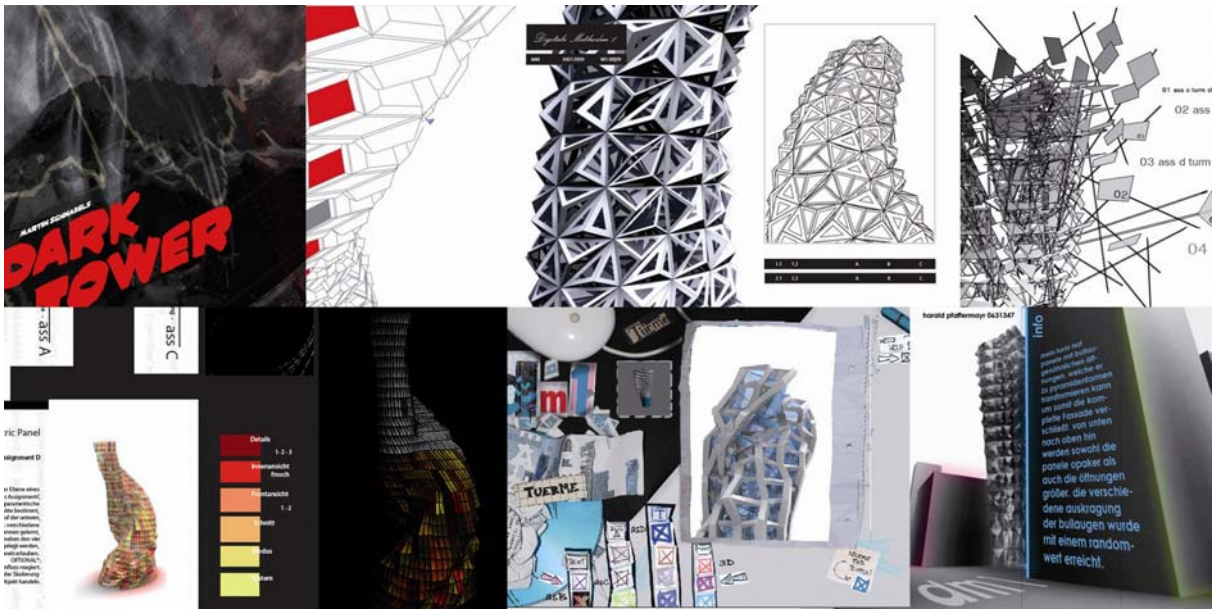

Fig. 7: Final assignment "Hyper-Portfolio" (Acrobat 3D).
Interactive, three-dimensional, cross-platform presentation.

References

[1] Negroponte, N. (2000). Pinstriped suits. In Maeda@Media. VII-V. New York: Rizzoli.
[2] Bell, D. (1976). The Coming of Post-Industrial Society. New York: Harper Colophon.
[3] Maeda, J. (2001). Design By Numbers. Cambridge: MIT Press.
[4] Dank, R. (2006). ryugyong.org. Master thesis, Graz University of Technology.
[5] McCloud, S. (1993). Understanding Comics. New York: Kitchen Sink Press.
[6] Kiesler, F. (1949). Manifested du Corréalisme. In L'Architecture d'Aujourd'hui. Paris: Mont-Luis.
[7] Bogner, D. (1997). Inside the Endless House. In: Friedrich Kiesler 1890 - 1965. 9-29. Vienna: Boehlau.
[8] Autodesk. (2010). MEL and Expressions. In: Maya Online Help. http://download.autodesk.com/us/maya/2010help/index.html?url=WS73099cc142f48755-4bc38c931187aa80bc9-32a7.htm,topicNumber=d0e156749 (accessed May 10, 2010)
[9] Schmitt, G. (2001). Creative Collaboration. In: Bits and Spaces. 36-39. Basel: Birkhaeuser.
[10] Cache, B. (2010). After Parametrics? In: GAM06: Nonstandard Structures. 50-61. Vienna: Springer.

Links

TUERME.n website (German) http://iam2.tugraz.at/dm1/w08/
Digital Design Methods website (German) http://iam.tugraz.at/dm2/
Institute of Architecture and Media website (German) http://iam.tugraz.at/