Corneel Cannaerts

Mediality of Code

Architectural Design and Coding Practices

Code Matters

The environments in which we operate as architects are increasingly saturated with digital technologies: global communication and transportation technologies, internet-of-things, mobile devices, increased satellite coverage, location-based services, ubiquitous computing, etc. Digital technologies have become so pervasive that the digital is noticed only by its absence, not its presence.[1] Digital technologies are affecting our environments across scales: on a planetary scale they are rewiring the technosphere[2] into an emerging accidental megastructure[3] giving rise to radically new social, political, economic and technological geographies.[4] Several dichotomies at the basis of architecture as a discipline are rendered useless through the pervasive impact of digital technologies: collapsing differences between city and periphery, between artificial and natural,[5] between public and private,[6] between mediated and bodily experiences.[7] Architecture seems to have a hard time responding to these changes and the spatial and material challenges they pose. Digital technologies like social media, are eroding the role of architecture as the main means of socialisation.[8] Architecture seems to be shifting from being part of the apparatus that forms societies, to being impacted by societal forces beyond its control, it is shifting from the program to the programmed.[9]

The most direct impact of digital technologies can be seen in the design media, tools and procedures of architectural practice: all phases of the design process involve the use of computation in some form or another: from ideation, schematic design, design development, over fabrication and construction, to use, maintenance and occupancy studies. Architects have approached digitalisation of their medial practices initially as merely digital versions of well-known tools of drafting, modelling and rendering. However, recently architects seem to be realising the more profound cultural changes these technologies entail.[10] In particular the role of architectural drawing, as the discipline's foundational means of designing and producing architecture, is deeply

1 See Negroponte 1998.

2 The notion of the Technosphere was developed by Half 2014.

3 See Bratton 2015.

4 See Mattern 2017.

5 See Young and Unknown Fields Division 2016.

6 See Gleiter and Steinert 2014.

7 See McCullough 2013.

8 See Gleiter and Steinert 2014.

9 See Rushkoff 2011.

10 See Gänshirt 2007.

affected by the shift from representation to simulation resulting from the digitalisation of design media.[11] The call for this edition of the journal identifies the medial practices of architecture from two points of view: firstly, the influence of design media on the design and production of architecture, secondly the impact of architecture on societies and users at large. This paper focuses mainly on the former, on the mediality of coding in architectural design. In response to the second point of view it seems more fruitful to reverse the directionality of the question, instead of questioning architecture's impact on users and societies, it is more fruitful to look into the impact of the increased digitalisation of our societies on the medial practices of architecture. As such it does not directly deal with the pervasiveness of digital technologies and their material, spatial and environmental impact, nor with architecture's role in the societies, cultures, and media ecologies it gives rise to. However, these phenomena demonstrate that code matters, that code is increasingly entangled within our environments and stresses the urgency for architects to engage with coding. It frames the medial practices of coding in architectural design within a larger understanding of architecture engaging the complexities of our contemporary world.

11 Scheer, David 2014 argues that the medial practices of architecture are going through a fundamental shift from representation to simulation.

Coding Cultures

> "Code is not purely abstract and mathematical; it has significant social, political, and aesthetic dimensions. The way in which code connects to culture, affecting it and being influenced by it, can be traced by examining the specifics of programs by reading the code itself attentively." (Montfort et al 2013: 3)

Code is cultural, it is authored and designed and the relationship between code and our design cultures are profound. While code is still considered a subject of research for engineers and computer scientists, an increasing interest can be noticed in the humanities, the arts and design, as is showcased by the emerging fields of digital humanities, software studies, critical code studies and media archaeology.[12] Architectural history and theory might benefit from a critical reading of code used by architects in order to comprehend the appropriation of digital technologies in architecture. Recently such an archaeological approach to research into the recent past of digital architecture has been developed through the exhibitions and publications *The Archaeology of the Digital* [13] and *When is the Digital in Architecture?* [14] In order to reflect on the mediality of coding in architecture this approach is especially promising because it closely examines the actual code, data, software and interfaces used by practitioners in addition to artefacts, publications and discourse.

12 See respectively, Drucker 2016; Fuller 2008; Marino, Mark 2006, and Parikka 2012.

13 See Lynn, Greg et al. (2013).

14 Goodhouse, Andrew et al. (eds) (2017).

Throughout the history of the impact of digital media on architecture, there have been architectural practices engaging the digital beyond the black box of pre-packaged technologies, exploring the algorithmic processes directly through scripting, parametric modelling, or programming custom design tools

from scratch. The history of the development of digital technology, especially the development of design software[15] provides essential research material for anybody interested in coding as an artistic or design practice. The work of pioneers such as Ivan Sutherland, Ted Nelson, Douglas Engelbart, Alan Kay and Nicholas Negroponte [16] and more recent work by people like Robert Aish,[17] or David Rutten [18] has made a substantial contribution to design culture. This contribution is situated more in providing the tools and frameworks for architectural design rather than pursuing architectural design practice itself through coding. In the recent past, there have been a number of key practitioners that do operate in this intersection of coding and architectural design.

In order to reflect on the mediality of coding in architecture, we will give an overview of how coding has been appropriated within architectural practice and the plural cultures of code it has given rise to. The goal of this selection of references is not to provide a comprehensive historic description of coding in architecture, but to highlight the plurality of coding cultures and the mediality of coding in architectural practice, as a foundation for reflecting on the contemporary situation.

The pioneering work in creative applications of computation can be traced back to the 1960s and 1970s, although most early creative applications of computation were in the field of computer arts, graphic design and animation, there are some examples within architecture. Some notable examples are John Frazer's *Reptile Structural system from 1966*[19] and his collaboration with Cedric Price on the *Generator Project*.[20] Pioneer in computer art Georg Nees, also experimented with architectural drawings [21] and collaborated with architect Ludwig Rase on the *Kubo-Oktaeder* project and the *Siemens Pavilion* at the Hannover Messe of 1970.[22] Those pioneering examples had access to computation through university main-frames, relying on custom software developed in collaboration with computer scientists. The computation happened far removed from the design process lacking direct feedback and often relied on pen plotters to render the results visual, as such coding was a distant medium far removed from the direct design ideation. Nevertheless, the computer played an active role in developing the architectural ideas of these projects, i.e. they are not singular architectural proposals but architectural systems that encode the relationship between the constituent parts and the overall whole.

The development of workstations and personal computers and CAD software during the 80s and 90s led to a gradual but widespread adoption of CAD in architectural practice, mainly as a digital version of drafting. Simultaneously, we see an interest in the potential of computation in architecture in avant-garde practices, academia and architectural theory. Coding is being used by more avant-garde architects like Shoei Yoh, Peter Eisenman, Frank Gehry and Chuck Hoberman using computation to explore questions on structural optimisation, executing formal operations, modelling complex geometries and kinetic structures.[23] Most of these projects use custom developed software, either by the architects themselves or through collabora-

15 For a good overview of both history and the present day situation, see: Manovich 2013: 39–43.

16 See Moggridge 2007, and Manovich 2013: 40.

17 See Aish 2003: 243–252.

18 See Davis and Peters 2013: 124–131.

19 See Frazer: 1995: 68–73.

20 See Furtado Cardoso Lopes, Gonçalo Miguel 2008: 55–72.

21 See Nees 1969 plotter drawing Hall generated between 1965 and 1968, Miguel 2008: 55–72.

22 See http://www.heikewerner.com/kubo_en.html, consulted on 15/11/2019, a description of the design process of the Siemens Pavilion can found in Vrachliotis 2011.

23 Greg Lynn explicitly contradicts the idea the early digital architecture of that period were happy accidents, resulting from architects playing around with software borrowed from other industries without clear understanding or design intent, see Lynn et al. (eds) 2013.

tions with computer scientists, or borrow software from animation and other industries to overcome the limitations of standard CAD software. Towards the early 90s architects started to use 3D modelling environments[24] to design and visualize architectural projects. The screen-based design media inspired architects to design mediated, virtual and interactive environments, often in collaboration with artists and designers at media arts institutes and festivals.[25] For architects such as NOX, dECOi architects, Assymptote, ONL, Bernard Cache and others, the experience of digital media itself became the subject of their design practice, as such coding was used not only as a design medium but also as a means of orchestrating interactive environments, the mediality of digital media became the actual materiality of the resulting architecture.[26] Towards the 90s digital design media were adopted widespread across architectural design practice, consolidating their position within the architectural toolbox, affecting all aspects of architectural design from ideation, design development, to detailing and production. The increased computational power and fidelity of design software resulted in architects developing projects with an unseen complexity affording architects control over their design and production.[27] The developments in computer hardware and software in early 00s led to a democratisation of access to computation, in architectural practice and academia. Computational design was made accessible through scripting interfaces and the development of computational design software specifically aimed at architects, designers and artists.[28] In the same period, we saw digital fabrication technologies becoming increasingly accessible within architectural education and practice: The FabLab project initiated at MIT by Neil Gershenfeld[29] aimed at opening up fabrication technologies to students, designers and the general public, encouraging sharing, and resulting in a global network of FabLabs. The *RepRap* project, initiated by Adrian Boyer of the University of Bath in 2005, which stands for replicating rapid prototyper, an open-source 3D printer able to produce most of its own parts.[30] Gramazio and Kohler started experimenting with robotic fabrication in 2005, using an industrial robotic arm to stack bricks.[31] Through these evolutions materiality and making became the focus of architects engaging the digital, this so called 'material turn' in digital architecture also reflected in a number of publications[32] and exhibitions[33] that emerged around the same time. Coding became an important means of interfacing with digital fabrication technologies bridging the digital and the material: fabrication properties and constraints are encoded within algorithmic processes. The resulting materiality of the fabricated artefacts depends on materials used, the fabrication machines and the code that runs them, in other words the mediality of coding manifests itself materially through digital fabrication processes.

Looking back on the recent history of the intersection of design and computation has the advantage of being a reflection from a certain distance in time; dissecting the cultures of code today is a much more difficult task. Digital technologies have democratised, not only the software and hardware for design, but also the means of spreading design ideas. Computational design

24 Catia was developed from 1981, Autocad introduced 3D modelling features in 1987, Softimage was developed in 1987 and Form*Z in 1989, a predecessor to Autodesk 3DS max was released in 1990, Rhinoceros3D and Maya in 1998.

25 Most notable are Ars Electronica Center in Linz, V2 institute for unstable media in Rotterdam, and Zentrum für Kunst und Medientechnologie ZKM in Karlsruhe.

26 See Lynn 2017: 300.

27 Ibid. 314.

28 Processing was initiated in 2001 (https://processing.org/), Generative Components in 2003 (https://en.wikipedia.org/wiki/GenerativeComponents), Grasshopper3D in 2007 (https://www.grasshopper3d.com/), accessed on 15/11/2019.

29 See Gershenfeld 2007.

30 See http://reprap.org/ (consulted on 20/09/2019).

31 See Gramazio and Kohler 2008.

32 Kieran and Timberlake 2004, Leach et al. (eds) 2004 and Kolarevic (ed) 2003.

33 The most notable exhibitions are the Architectures Non-Standard organized at the Centre Pompidou in Paris in 2003, and the Scripted by Purpose at Fuel Gallery in Seattle in 2007.

has spread out not only in its application, but also in its development and reception, it forms a diverse and layered landscape of positions, approaches and tools that can be compared to an ecosystem.[34] Design software has changed from large corporate-developed standalone applications to an array of custom tools, plug-ins, platforms and libraries often developed by practitioners missing certain tools for their needs. Notwithstanding the increased number of practitioners involved and different forms of exchange emerging in the form of online communities and networks, this landscape still seems to cluster around a relatively small number of schools, institutes and conferences.[35] While there is an extensive number of publications, not to mention videos, blog-posts and tutorials,[36] on the contemporary impact of digital technologies on architecture and design, few explicitly focus on coding, and those that do tend to take either the form of instruction books[37] or merely discuss the outcomes of computational design processes. Most instruction books include some examples of work and will hint at the reasons for engaging with code and reveal some of its cultural traits; likewise, publications focussed on works will provide some but limited insight into the codes behind the works, and few publications find middle ground between showing crucial work and acknowledging the role of code.[38]

Even fewer publications look into the cultural dimensions of coding in architectural design.[39] The most comprehensive account on the contemporary cultures of coding in architectural design is Mark Burry's *Scripting Cultures: Architectural Design and Programming*.[40] This book raises a number of questions on the role of coding in design practice: Why do designers and architects engage with code? What added value does scripting bring to the design process? What does scripting entail for collaborating in teams, and what are its implications for authorship in design? The book provides tentative answers to these questions by sketching a brief history of computation in design, providing a number of in depth personal accounts on scripting in design practice and teaching, and an inquiry of thirty highly renowned practitioners in architecture and design that use scripting as an important part of their practice. The overall image *Scripting Cultures* draws is one of scripting as a hard won skill and a highly diverse and articulated map of coding practices hence the plural coding cultures.

Encoded Design Worlds

> "Designers often establish design worlds implicitly, through their choice of media and instruments. A drawing board and traditional drafting instruments, for example, establish an Euclidean design world populated by two kinds of graphic tokens – straight lines and circular arcs – that can vary in size and position and be related to each other as parallels, perpendiculars and so on. A designer toying with cardboard working models enters a design world populated by plane polygons that can be shaped in different ways and translated and rotated in three dimensional space. De-

34 See Davis and Peters in: Peters and De Kestelier (eds) 2013: 124–131.

35 These following schools are prominent: AADRL, Bartlett, SCI-Arc, MIT media Lab, ETH, SIAL RMIT.

36 See for example http://designreform.net/, http://www.designalyze.com/ or http://www.plethora-project.com/ (consulted on 26/01/2018).

37 For example, Reas, Casey and Fry, Ben (2007), Greenberg, Ira (2007), the Grasshopper Pirmer by ModeLab see http://modelab.is/grasshopper-primer/ (consulted on 26/01/2015).

38 An exception to this is Reas, Casey and McWilliams, Chandler (2010). This provides an interesting and broad overview of works from the disciplines mentioned in the title, spanning the last 50 years, and is organised according to the principles the authors see as crucial traits of code – repeat, transform, parameterize, visualize and simulate – and for each they provide example code.

39 Two editions of Architectural Design provide insight in the cultures of code: Programming Cultures: Art and Architecture in the Age of Software (2006) and Computation Works: The Building of Algorithmic Thought (2014), although they both provide a quite selective overview of coding practices.

40 See Burry 2011.

signers shaping clay with their fingers or cutting polystyrene blocks with hot wires, enter yet other kinds of design worlds." (Mitchell, 1990: 38)

Architecture as a discipline has long understood digital technologies in terms of its own traditions; digital design tools were developed as digital versions of the well-established practices of drafting, modelling and rendering. The choice of medium and instruments establishes what Mitchell calls a design world, which defines both the basic tokens that can express design ideas and the possible transformations of them. The idea of the design process being acted out in a *design world* frees design media from being passive recipients for design ideas to being active contributors to the design process and suggests exploration rather than closure. In other words, the design process can be described as the gradual development of design ideas, by the architect through exploring design worlds established by the choice of media, a negotiation where neither designer nor the design media predetermines the outcome.

Notwithstanding the similarities between analogue and digital versions of drawing, modelling and rendering, as design worlds they are fundamentally different.[41] In his description of design worlds, Mitchell describes modelling with cardboard as a design world populated with planar polygon shapes. However, possible manipulations of cardboard are not limited to just cutting, the resulting forms are not confined to geometric categories such as planar or polygonal, and cardboard comes with additional material properties such as texture, colour and weight. Analogue design media are inherently open both in the tokens they allow and operations that act on them, furthermore they are subject to unforeseen influences from the environment in which they are deployed. Analogue design media, in particular physical models, behave *analogous* to the material artefacts they refer to, a piece of cardboard or paper, no matter how light, will not float in thin air.[42] In contrast digital design media only work with computable, numeric or geometric entities, and the manipulations are limited to a preprogrammed finite set of options. In other words, digital design worlds are closed both in the tokens it contains and the operations that act on these tokens. Digital design media operate on a finite set of computable elements and thus require design ideas to be described in numeric and geometric terms. To digitally describe continuous spatial and material phenomena they are sampled at discrete intervals at a specific level of detail or resolution.[43] The fundamentally discrete nature of digital media introduces a kind of *digital materiality*,[44] rather than analogues, this operates rather differently than the material it refers to. As architectural practice and the environments it operates in become more mediated through digital technologies these *digital materialities* are increasingly manifesting itself.

In general architectural practice digital design media are used through hardware and software interfaces, these expose functionality through textual commands and graphic symbols while their actual algorithmic operation remains hidden. Exploring the digital design worlds established by design media happens through predefined hardware and software, that does not expose the

41 See Aicher; Vossenkuhl and Robinson 2015.

42 Selenitsch 2007: 8.

43 Menkman 2011.

44 See Gramazio and Kohler 2008.

actual computational tokens or operations or algorithms. The argument here is not that using design software limits what can be designed, black boxing does not necessarily limit what can be explored in a digital design world, but the *mediality* of that exploration. Furthermore, predefined digital media can be used, or even abused[45] in ways not intended by the developer to reach design results beyond the foreseen outcomes. This can range from abusing existing features of software to reach different results, to breaking protocols by working on other file formats than intended or abusing design flaws or bugs in the software to deliberately break or hack its operation.[46] As such digital media just like their analogue counterparts might provide tokens and operations beyond the predefined through glitches[47] or hacking. Through breaking the design world, errors or glitches, whether accidental or intentional, partly reveals the mediality of the digital design medium, exposing the tokens and algorithms and the resolution at which it operates.

45 See Glanville 1992: 214.

46 See Cannaerts 2020.

47 See Menkman 2011.

Through the development of scripting interfaces, parametric modelling, or programming custom design tools from scratch, contemporary creative practices can engage the digital beyond the black box of pre-packaged technologies. This allows for exploring tokens and algorithmic operations directly, making the *mediality* of coding part of the design process. However, we can distinguish between the role computation itself plays within design processes mediated through digital media. In *Algorithmic Form*, Kostas Terzidis[48] draws a distinction in architectural design between *computerisation*, where existing concepts are simply stored and manipulated using computer technology, and *computation*, where algorithms actually contribute to the formation of design outcomes. Terzidis sees code as an extension of human thought, which is fundamentally different, or what Terzidis calls *allo*, derived from Greek, meaning other. Terzidis sees algorithmic computation not as an extension of human cognition, but as a fundamentally different form of cognition. In *Programming Architecture*, Paul Coates[49] makes a similar distinction between code as data that merely stores information and code as a programming language, as a text that can read and alter itself, where there is no distinction between data and instruction.[50]

48 See Terzidis 2003: 65–73.

49 See Coates 2010: 3.

50 Coates bases this argument on Friedrich Kittler, "Code or How to Write Something Different" (1998) indicating that this is an essential property of a Turing Machine.

The ability of architects to go beyond the black box of existing digital design media, to directly manipulate the tokens and algorithms that drive a design result through coding, allows for opening up the closed nature of the design worlds as established by digital design media. The practices of coding allow for actively engaging with the mediality of digital design media, where design processes become a negotiation between the designer and the agency found in code.

Coding as Creative Practice

The published research and discourse on the deeper engagement with coding in architectural design, discuss design processes, specific algorithmic procedures and fabrication techniques, as well as the resulting architectural designs, projects and artefacts. However, the mediality of working with digital

design media on an algorithmic level is seldom discussed, i.e. what coding as a medium contributes to the design process and outcomes and how that differs from other design media. In the following section I will attempt to unravel some aspects of the mediality of coding, based on my personal experience of using coding in architectural design processes. I have used coding extensively in creative practice and teaching for over a decade, the output ranging from on screen graphics, video and animation, objects and interactive installations, to fabricated artefacts and prototypes to architectural projects. I have used both text based and graphical programming in a plethora of languages, ranging from extending existing software through scripting, interfacing with various hardware devices, to writing custom design tools from scratch.

This approach is in line with recent developments in architectural research, mostly described as research by design and creative practice research.[51] In this mode of research, which combines action and reflection, practice and theory, the act of designing is a substantial and crucial driver for research. The research discussed in this section uses this mode of inquiry, which can be described as the research into the *medium itself*,[52] as it looks into the role of coding as a design medium in creative practices of architecture and design. Coding is approached not with the aim towards the fundamental understanding of a computer scientist or the professional understanding of a software developer, but rather with the aim of exploring its potential for design practice. The aim here is to give an in-depth account and highlight several aspects of the mediality of coding, however this does not cover every aspect of how coding can inform design processes and might not be applicable to other creative practices.

51 See Verbeke 2013: 137–160. Verbeke (et al.) 2015: 25–28.

52 See Blythe and van Schaik 2013: 53–69.

53 It is explicitly mentioned as a reason to explore generative systems by John Frazer and Paul Coates and some of the respondents of Mark Burry's Scripting Cultures. See Burry 2011: 28.

Exploring Design Variation

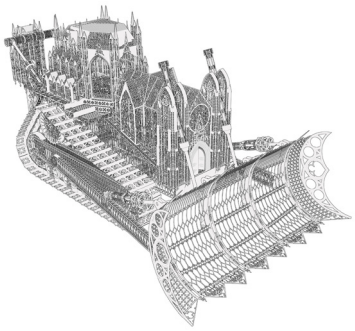

Fig. 1 Gothic Sculptures Design Model

As indicated by many of the practitioners in Mark Burry's questionnaire,[53] an important incentive to start coding is avoiding repetitive work by automating design procedures. Similarly, my initial experience with using coding in design practice stems from automating the design and fabrication processes while designing *Gothic Sculptures* for visual artist Wim Delvoye (fig. 1). This was done through chaining often used commands into macro-commands, and through AutoLisp scripts facilitating the fabrication of the sculptures. Next to avoiding repetitive work, optimising processes of fabrication and making were important reasons for integrating coding in my design practice. While chaining macro commands and scripts to avoid repetitive work might seem modest, this is what coding essentially entails: defining a set of precise instructions for the computer to execute. Introducing coding into a design process implies a shift from addressing a design problem one command at a time to defining a chain of instructions, whether it takes the form of a macro command or script working on top of a modelling application, an associative parametric model or writing code from scratch.

While using code to avoid repetitive work is probably applicable to other designers and architects,[54] it also tells us something about the nature of com-

54 See Burry 2011: 28.

putation itself. Computers "are designed to accurately perform the same calculations over and over"[55]; repetition can be called the computer's unique talent. Just as essential to computation is variation, as embodied by the variable. Computation decouples the logic of the algorithm as a precise set of instructions from the specific numeric instances it operates on. Although algorithms are inherently repeatable, each iteration can be different, based on variables. This tension between repetition and variation has not only been used to increase productivity but has also been explored as a quality by many practitioners using code in their art or design practice. It is a theme prominent in the work of early practitioners in computer arts such as Georg Nees, Frieder Nake, Vera Molnar and Manfred Mohr as discussed above. Designing through coding shifts the attention of the designer from the singular design solution to the logic that drives a particular design question or potential, it provides a non-linear way of exploring design variation.

55 Reas et al. 2010: 53.

Sketching with Code

Since the adoption of computers in architectural practice there has been a debate on whether CAAD can be used as a sketching medium. The long held belief that design ideation was only possible through hand drawing[56] and CAAD was limited to refining and documenting already existing design ideas is challenged by more recent research.[57] Furthermore, Schaeverbeke and Heyligen-[58]argue that in contemporary practice architects use different design media simultaneously or switch between digital and analogue media. Both analogue and digital media can be used for sketching if designers are fluent enough, and design ideation can be located in this 'in-between'.[59] The appropriation of coding by architects has to be seen in light of this fluid and diverse use of media in design practice, it cannot be seen in isolation but is adding another layer to the multimodal feedback of design processes.

56 See Schön 1983, Garner 1992: 98–109, Goel 1995 and Bilda and Demirkan 2003: 27–50.

57 See Jonson 2005: 613–624.

58 See Schaeverbeke and Heylighen 2012: 49–57.

59 Ibd.

In *Tools for Ideas* Gänshirt [60] makes a distinction between verbal and visual design tools, corresponding with complementary ways of thinking: verbal, linear, logical thinking on the one hand, visual, spatial, associative thinking on the other. Based on McLuhan's[61] thesis that media are ordered hierarchically in terms of abstraction, coding can be seen as an abstract verbal design medium, whereas sketching is classified as an intuitive visual medium. Interestingly enough Gänshirt[62] defines the computer as being on the intersection between visual and verbal tools. Even though writing code is a verbal design tool the feedback it provides to the designer is partly visual, and if the designer is fluent enough in writing code it can be used as a sketching tool.

60 See Gänshirt 2007.

61 McLuhan 1964: 22.

62 Gänshirt 2007: 101.

This notion of sketching, bridging the gap between verbal and visual and exploring design ideas through writing code is central to Processing, an open-source programming language, development environment and community of creative coders.[63] Programs written in Processing are called sketches, and they are collected in a sketch book. Its syntax refers to terms used in design practice and software: drawing, stroke, fill, etc. Its simple interface, reference and syntax is deliberately designed to reduce the gap between writ-

63 Maeda 2004; Reas and McWilliams, Chandler 2010; Reas and Fry 2007.

ing code and getting visual feedback on the screen. Its core functionality can be extended, with third party libraries developed by the community, into a means for prototyping and production.

In my design practice sketching through writing code plays an important role, I have used Processing extensively ranging from quick sketches, design experimentation, coding tools for other architects, designers and artist, writing code for specific fabrication machines.[64] The sketches shown in figure 2, can be compared to doodles, or explorations of specific generative systems. Moments of playing, interacting with the graphical representation of code and tweaking values are alternated with changes to the code itself. Rather than describing the specific details of these projects, in this section of the paper I reflect on the experience of sketching with code and how its mediality informs my design practice.

64 Cannaerts 2015a and Cannaerts 2015b: 152–161.

Working with code as a design medium provides the designer with different kinds of feedback on the screen: a graphical window showing the result of the running code, a textual one showing the actual code itself and possibly textual feedback through the console. Although the running code can be made to respond to various inputs, for example mouse and keyboard, the design mainly progresses by working on the code itself. Text-based coding is an unforgiving medium, forgetting even one character will lead to a syntax error, and it is often hard to tell from the visual feedback alone what is exactly going on in an algorithm. These limitations can be overcome by continuously testing the code, incrementally building on working versions of the code and using the console to provide textual feedback, or by developing a debug mode that renders certain information on the screen. Graphic coding interfaces tend to be a bit more forgiving, as code is contained within blocks with clearly defined inputs and outputs, but they tend to become quite hard to read once definitions get larger.

As stated above, sketching with code adds another layer to the multimodal interaction characteristic of design processes, and does not replace sketching with pen and paper but rather complements it. While coding, sketching on paper can greatly help with visualising ideas while simultaneously testing them out in code. Different from sketching with pen and paper, sketches with code develop incrementally, not by retracing the sketch, but by incrementally building on previous blocks of code. The reuse of code and gradual increase in complexity allows sketches to be turned into blue prints or prototypes and even actual design projects.

Code as a medium to develop ideas tends to progress in chunks as parts of the underlying algorithm get defined, evaluated and refined. Developing an algorithm often happens through a simplified version of the design problem at hand, which can take the form of simpler input geometry or a low number of iterations or variables. Once an algorithm reaches a certain state of development, I tend to increase the complexity, which is often a revealing moment. Both text based and graphical programming environments provide a layered feedback, design progresses by working on the geometrical and the algorith-

mic simultaneously. This layered feedback, giving insight in the geometric and algorithmic at the same time, allows for developing an understanding of the logic underlying a certain design problem.

Making and Using Tools

Using code as a design medium affords control over the algorithms beneath the interface of software tools and, as such, allows designers to develop their own design tools. Going beyond the intended use of a tool or developing your own tools is frequently mentioned by practitioners as a main motivation for using code as part of their creative practice.[65] In architecture, this is a prominent argument in many publications on parametric and algorithmic design.[66] This position is most explicitly stated by Aranda and Lash.[67] Under the title *Tooling*, they describe a number of algorithmic techniques which are illustrated by a version of the algorithm in pseudo code, a number of experiments and a project developed with this specific algorithm.

In 2013 I gave a workshop together with Phil Ayres and Hollie Gibbons at CITA in Kopenhagen to start the Adaptive Aggregates installation project as the start of the CITA studio master program. The workshop was conceived as an introduction to computational design that required no prior knowledge of coding. Three design tools were developed looking into simulating aggregate structures across a number of scales. The design tools used text-based coding in Processing and parametric modelling in Grasshopper, while they could be used without altering the actual code through user interfaces, they introduced various computational design strategies and provided the source code for students willing to learn how the underlying code worked.

The first tool (fig. 3) looked into the fabrication and inflation of the material component, it was developed through iterative material testing and simulated using a particle spring model in Grasshopper and Kangaroo. The components consisted out of an inflated tube, were fabricated prior to the workshop in three scales. The second tool (fig. 4) was coded in Processing and explored different ways of connecting the components, it allowed for exploring branching structures based on the logic of connecting the different components. The user interface allowed for control of the geometry of the components, the rules of connection, the noise introduced by small deviations in the connecting system. The third tool (fig. 5) used a rigid body simulation to simulate the pouring or packing aggregations, the user interface allowed for the position, type and number of components to be poured into the scene, an environment mesh could be used to simulate pouring over different objects and obstacles.

In order to overcome the limits inherent in design software, we used different computational design tools and strategies to develop a spatial proposition. We chose to operate simultaneously in different software environments and use different computational design techniques: explicit geometric modelling in Rhino, visual parametric modelling in Grasshopper and text-based scripting in Processing. The preparation resulted in three design tools; com-

65 Maeda 2004.

66 Kilian 2006 and Meredith (ed) 2008 and Burry 2011.

67 Aranda and Lash 2005.

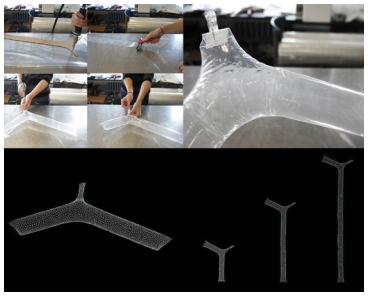

Fig. 3 Tool 1: Material and simulated inflation

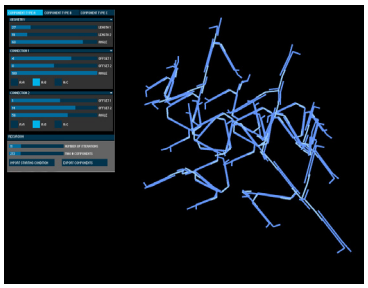

Fig. 4 Tool 2: Branching based on connecting logic

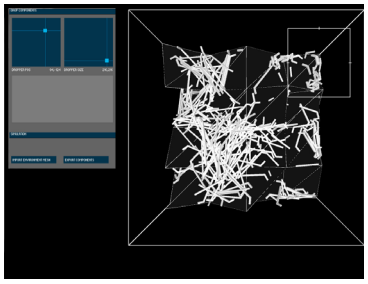

Fig. 5 Tool 3: Pouring simulation

putational techniques were made available through design tools with graphical interfaces. The position, orientation and geometry of the components could be described by the three coordinates and an offset thickness, and all the tools could export and import text files containing this information. Rhino was central in the work-flow; it was used to collect and compile the multiple explorations, prepare the starting conditions and environment meshes as input for the simulations and assemble the different strategies into a final design proposal.

In the tools coded for *Adaptive Aggregates* workshop, simulations could be found on multiple scales. Through nesting different simulations, it becomes clear that each tool comes with its own assumptions and its own requirements and limitations and all provide a different insight in the design task at hand. All of these simulations have their affordances and capture materiality in discrete encoded elements, from the particles and springs simulating the inflation to the compound shapes assembled out of convex parts in the rigid body simulation. Furthermore, time is encoded as explicit discrete steps: the algorithm that makes up the simulation computes the resulting world one iteration at a time.

Having access to the code that drives software tools can allow for a deeper understanding of the design issues at hand and uncover the assumptions inherent in the tools. Actively developing this code allows for these assumptions to be questioned and explored differently. In the *Adaptive Aggregations* workshop, I was often reminded of my own assumptions as students were using the tools. To that extent, the tooling metaphor is useful, but it also introduces an opposition between tool making and tool using, which in my practice of using code as a design medium are not separate activities, rather they mutually inform each other. The process of making a tool gradually unravels the design problem at hand. Furthermore, coding your own design tools does not generally start from a blank canvas but is instead based on examples, code snippets, add-ons and libraries often developed by others. The environments in which you code are obviously tools themselves, with their own assumptions, limits and potential, programmed by someone else.

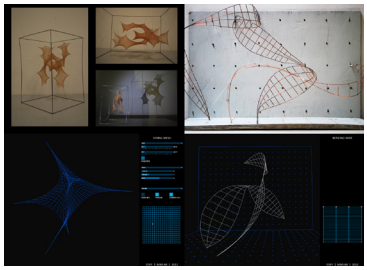

Fig. 6 Material & Digital Form Finding Workshop

Structuring Code

In the *Material & Digital Form Finding* workshop (fig. 6) material experimentation formed the basis for coding a number of bespoke design tools that allowed for exploring the spatial potential of form finding systems. Conceptually translating material system into code requires decisions into how to structure the code, defining what the material components are and how do they relate to computational concepts. In this project they were simulated as particle spring-models,[68] where springs and particles are modelled as autonomous objects negotiating their environments, the forces at play, and other components.

Much of the syntax of code has to do with structuring repetition and meaningful variation. In text-based coding this is reflected in defining and declar-

68 Kilian 2006.

ing variables, loops, conditionals, functions, objects, classes etc., which are all means of efficiently structuring code and determine the flow of instructions passed to the computer.[69] The elements for structuring code are highly hierarchical and are geared towards modularity and reuse, and splitting up a design problem into reusable chunks.

Findings from the *Material & Digital Form Finding* workshop were further developed. The *Folded Strips Pavilion* project extended the relationship between material and digital to be more iterative and cyclical by also including digital fabrication. Since the project required digital fabrication the workflow was based on Rhino and Grasshopper as a more robust modelling environment, which was further extended with Kangaroo, a physics simulation add-on for Grasshopper and Anemone, which allows recursion within the Grasshopper environment. The *Folded Strips Pavilion* used a physics simulation to derive the overall form of a pavilion (fig. 7). The design simulated a particle-spring model based on hexagonal grid, the grid can be deformed to allow for denser or less dense areas. In order to be fabricated the data tree of the hexagonal grid is reorganised as continuous folded strips rather than hexagonal cells.

In visual programming languages, such as Grasshopper, code is structured as a network of components, where each component computes an output based on data it receives as an input. Different than in text-based programming, the flow of execution is explicitly visualised, which provides a clear feedback of the algorithm. When definitions become more complex, components can work on lists of data and nested lists of data, or data trees, it becomes increasingly harder to understand what is happening in the algorithm which necessitates clearly structuring the definitions through grouping, naming and clustering.[70]

The way code is structured through text based code or visual programming languages provides a framework for architectural design thinking, these are not neutral and will influence what architectural ideas can be designed and produced.

Conclusion

As digital technologies are increasingly influencing the environments in which we operate as architects, engaging with coding in the medial practices of architecture becomes necessary, for its impact on the design media architects use and for interfacing and making sense of the technologically saturated world.

Tracing the history of the adoption of code in creative practice reveals various ways in which the mediality of code manifests itself in the design and production processes of architecture. From the pioneering engagement of code as means of rethinking architectural processes and structural systems of the 60s and 70s; over the conceptual and formal explorations of avant-garde practices in the 80s; moving beyond the static and material limitations of architecture through experimentation with virtual, kinetic and interactive environments of the 90s, where mediality of digital design media becomes the ar-

69 Fry 2008.

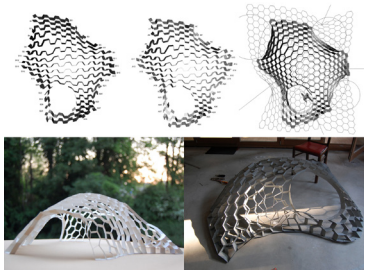

Fig. 7 Folded Strips Pavilion: Design Model, Fabrication Model, Paper & Metal Prototype. Workshop

70 Davis, Burry, and Burry, M.: 2011: 55–68.

chitectural project; the material manifestation of mediality of coding through digital fabrication in the 00s; to the plethora of manifestations of mediality in contemporary practice. In all of the discussed examples the digital design media and the encoded design worlds give rise to play an active role in contributing to the design process. Coding allows for design explorations beyond the initially closed design world established by digital design media, by providing access to the tokens and algorithmic process.

While the metaphors used in the last section of the paper to describe coding as a creative practice—exploring variation, sketching, tooling, structuring each highlight a particular aspect of how coding operates as a design medium, none describes the use of coding in all its intricacy. Because of its re-usability and fluidity, code as a design medium can switch between these different modes of informing the design process. The cases discussed in this paper demonstrate that code is a specific design medium with its own affordances and resistances. In other words, code has agency in the design process and each of the discussed metaphors unveils part of that agency.

The fundamental difference between coding and most other design media, is the layeredness of the provided feedback. Coding allows for simultaneously working on the geometry representing a particular instance of the design and the textual representation of the logic that drives that design. Through having sufficient experience in coding and using intuitive interfaces, coding can become a form of sketching, although it operates through both textual and geometric manipulations. Working on verbal and visual representations of design is also reflected in the difference between making and using tools. Instead of promoting the first over the latter it is through the switching between both that designing with code progresses. Both object-oriented programming in Processing and data trees in Grasshopper are elegant structures, computationally but also in visualising the structure of the algorithm, this paper demonstrates that those structures can be a conceptual driver for design.

Author

Corneel Cannaerts is an architect, lecturer and postdoctoral researcher at the KU Leuven Faculty of Architecture interested in the impact of emerging technologies on the culture and practice of architecture.

Literature

Aicher, Otl (2015): Analogous and Digital. Berlin: Ernst & Sohn.

Aish, Robert (2013): "Extensible Computational Design Tools for Exploratory Architecture," in: Kolarevic, Branko (ed.): Architecture in the Digital Age: Design and Manufacturing. New York: Spon Press: 243–252.

Aranda, Benjamin and Lash, Chris (2005): Tooling. Pamphlet Architecture 27. New York: Princeton Architectural Press.

Bilda Zafar and Demirkan Halima (2003): "An insight on designer's sketching activities in traditional versus digital media," in: Design Studies, 24: 27–50.

Blythe, Richard and van Schaik, Leon (2013): "What if Design Practice Matters?" in: Murray, Fraser (eds.), Design Research in Architecture: An Overview. Burlington: Ashgate: 53–69.

Bratton, Benjamin (2015): The stack: on software and sovereignty. Cambridge: MIT Press. The stack: on software and sovereignty. Cambridge: MIT Press.

Burry, Mark (2011): Scripting Cultures: Architectural Design and Programming. London: Wiley.

Cannaerts, Corneel (2020): "Hacking Agency: Digital Fabrication as Design Medium," in: Barlieb Christophe, and Gasperoni, Lidia (eds.), Media Agency – Neue Ansätze zur Medialität in der Architektur. Bielefeld: Transcript Verlag: 179–196.

--- (2015a): Negotiating Agency: Computation and Digital Fabrication as Design Media, PhD dissertation, RMIT University.

--- (2015b): Fabricating Material Intensities: Designing and Making Mediated by Computation. Making Research | Researching Making. Creative Practice Conference. Aarhus: Arkitektskolen Aarhus: 152–161.

Coates, Paul (2010): Programming.Architecture. London: Routledge.

Davis, Daniel, Burry, Jane, and Burry, Mark (2011): "Untangling Parametric Schemata: Enhancing Collaboration through Modular Programming," in: Designing Together: Proceedings of the 14th International Conference on Computer Aided Architectural Design Futures, ed. Pierre Leclercq, Ann Heylighen, and Geneviève Martin, Liège: Les Éditions de l'Université de Liège: 55–68.

Davis, Daniel (2013): "Modelled on Software Engineering: Flexible Parametric Models in the Practice of Architecture". PhD dissertation, RMIT University.

Davis, Daniel and Brady, Peters (2013): "Designing Ecosystems, Customising the Architectural Design Environment with Software Plug-ins," in: Brady Peters and Xavier De Kestelier (eds.), Computation Works: The Building of Algorithmic Thought, London: Wiley: 124–131.

Fraser, Murray (2014): Design Research in Architecture: An Overview, Burlington: Ashgate.

Frazer, John (1995): An Evolutionary Architecture, London: Architectural Association.

Fry, Ben (2008): "Deprocess", art work, in: http://benfry.com/deprocess/ (January 5, 2016).

Fuller, Matthew (2008): Software Studies: A Lexicon. Cambridge, Mass: MIT Press.

Furtado Cardoso Lopes, Gonçalo Miguel (2008): "Cedric Price's Generator and the Frazers' Systems Research," in: Technoetic Arts a Journal of Speculative Research: 55–72.

Gershenfeld, Neil A. (2007): Fab: The Coming Revolution on Your Desktop – from Personal Computers to Personal Fabrication. New York: Basic Books.

Glanville, Ranulph (1992): "CAD abusing computing", in: Mortola, Elena et al (eds), CAAD Instruction: The New Teaching of an Architect?, eCAADe Proceedings. Barcelona: 213–224.

Gleiter, Jörg H. and Steinert, Tom eds. (2014): Architecture and Social Media, Wolkenkuckucksheim | Cloud-Cuckoo-Land | Воздушный замок, Vol. 19, Issue 32.

Goel, Vinod (1995): Sketches of Thought, Cambridge: MIT Press.

Goodhouse, Andrew (2017): When Is the Digital in Architecture? Montréal: Canadian Center for Architecture.

Gramazio, Fabio and Kohler, Matthias (2008): Digital Materiality in Architecture. Baden: Lars Müller Publishers.

Gramazio, Fabio, Matthias Kohler, and Jan Willmann, eds. (2014): The Robotic Touch: How Robots Change Architecture. Zurich: Park Books.

Haff, Peter. K. (2014): "Technology as a Geological Phenomenon: Implications for Human Well-Being", in: Geological Society, London, Special Publications, vol. 395, nr. 1: 301–09.

Jonson, Ben (2005): "Design ideation: the conceptual sketch in the digital age", in: Design Studies 26: 613–624.

Kieran, Stephen and Timberlake, James (2004): Refabricating Architecture: How Manufacturing Methodologies Are Poised to Transform Building Construction. New York: McGraw-Hill.

Kolarevic, Branko ed. (2003): Architecture in the Digital Age: Design and Manufacturing. London, New York: Spon Press.

Leach, Neil, Turnbull, David, and Williams, Chris (2004): Digital Tectonics. Chichester, West Sussex, U.K., Hoboken, NJ: Wiley-Academy.

Lynn, Greg, Peter Eisenman, Frank O. Gehry, Chuck Hoberman, Shoei Yoh, eds. (2013): Archaeology of the Digital, Montréal: CCA, Canadian Centre for Architecture.

Lynn, Greg (2017): "Going Native: Notes on Selected Artifacts from Digital Architecture at the End of the Twentieth Century," in: Goodhouse, Andrew, and Canadian Centre for Architecture, eds. When Is the Digital in Architecture? Montréal: Canadian Center for Architecture: 280–334.

Maeda, John (2004): Creative Code. New York, N.Y: Thames & Hudson.

Marino, Mark (2006): Critical Code Studies in Electronic Book Review. http://electronicbookreview.com/essay/critical-code-studies/ (November 10, 2019)

Mattern, Shanon (2016): "Cloud and Field, On the resurgence of "field guides" in a networked age," in: Places Journal, https://placesjournal.org/article/cloud-and-field/ (November 10, 2019)

Mattern, Shannon (2017): Code and Clay, Data and Dirt: Five Thousand Years of Urban Media. University of Minnesota Press.

McCullough, Malcolm (2013): Ambient Commons: Attention in the Age of Embodied Information. Cambridge: The MIT Press.

McLuhan, Marshall (1964): Understanding Media: The Extensions of Man. London: Routledge.

Menkman, Rosa (2011): "Glitch Studies Manifesto," in: Lovink, Geert, and Rachel Somers Miles, eds. Moving Images beyond Youtube. Video Vortex Reader, Amsterdam: Institute of Network Cultures, https://beyondresolution.info/Glitch-Studies-Manifesto (November 15, 2019).

Meredith, Michael (2008): From Control to Design: Parametric/Algorithmic Architecture. Barcelona: Actar.

Mitchell, William J. (1990): The Logic of Architecture: Design, Computation, and Cognition. Cambridge: MIT Press.

Montfort, Nick et al eds (2013): 10 PRINT CHR$(205.5+RND(1));:GOTO 10, Software Studies, MIT Press

Negroponte, Nicholas (1998): "Beyond Digital," in Wired, Issue 6.12, https://web.media.mit.edu/~nicholas/Wired/WIRED6-12.html (November 10, 2019).

Pickering, Andrew (1995): The Mangle of Practice: Time, Agency, and Science. Chicago: University of Chicago Press.

Reas, Casey and Ben Fry (2007): Processing: A Programming Handbook for Visual Designers and Artists. Cambridge: MIT Press.

Reas, Casey, and Chandler McWilliams (2010): Form+Code in Design, Art, and Architecture. New York: Princeton Architectural Press.

Rushkoff, Douglas, and Leland Purvis (2011): Program or Be Programmed: Ten Commands for a Digital Age. Berkeley, CA: Soft Skull Press.

Schaeverbeke, Robin, and Ann Heylighen (2012): "In Search of the In Between," in: Physical Digitality: Proceedings of the 30th eCAADe Conference, Prague, Czech Republic: Czech Technical University in Prague, Faculty of Architecture: 49–57.

Scheer, David R (2014): The Death of Drawing: Architecture in the Age of Simulation. London/New York: Routledge.

Schön, Donald (1983): The Reflective Practitioner. New York: Basic Books

Selenitsch, Alex (2007): "Small Real Large," in: Mark Burry, Michael Ostwald, Peter Downton and Andrea Mina (eds.), Homo Faber: Modelling Architecture. Melbourne: Spatial Information Architecture Laboratory and the Melbourne Museum: 4–10.

Silver, Mike ed (2006): AD: Programming Cultures: Art and Architecture in the Age of Software. London: Wiley-Academy.

Terzidis, Kostas (2003): Expressive Form. London: Taylor & Francis Group.

Terzidis, Kostas (2006): Algorithmic Architecture. Amsterdam/Boston: Architectural Press.

Verbeke, Johan (2013): "This is Research by Design", in: Frazer Murray, Design Research in Architecture. UK: Ashgate Publishing: 137–160.

Verbeke, Johan, Zupanzic, Tadeja, and Achten, Henri (2015): "Digital Tools and Creative Practice in Architectural Research", in: Martens, Bob et al (eds) Real Time, eCAADe Conference Vienna. Vienna: eCAADe: 25–28.

Vrachliotis, Georg (2011): Geregelte Verhältnisse. Zurich: Ambra Verlag.

Young, Liam & Unknown Fields Division eds. (2016): Tales from the Dark Side of the City. London: AA.

Figures

Figs. 1–7 Corneel Cannaerts.

Recommended Citation

Corneel Cannaerts
Mediality of code. Architectural design and coding practices.

In: Wolkenkuckucksheim | Cloud-Cuckoo-Land | Воздушный замок,
International Journal of Architectural Theory (ISSN 1430-0984), vol. 25, no. 40,
Media Practices of Architectural Design, 2021, pp. 25–43.