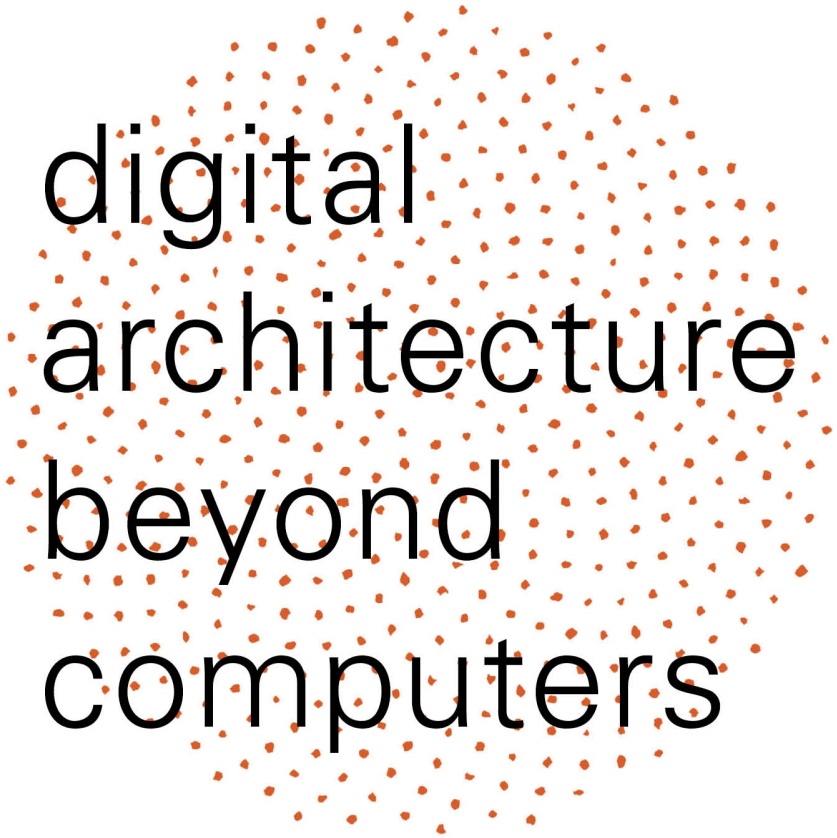
roberto bottazzi

fragments of a cultural history
of computational design



digital architecture beyond computers

BLOOMSBURY

Digital Architecture Beyond Computers

Digital Architecture Beyond Computers

Fragments of a Cultural History of Computational Design

Roberto Bottazzi

BLOOMSBURY VISUAL ARTS
LONDON • NEW YORK • OXFORD • NEW DELHI • SYDNEY

BLOOMSBURY VISUAL ARTS
Bloomsbury Publishing Plc
50 Bedford Square, London, WC1B 3DP, UK

BLOOMSBURY, BLOOMSBURY VISUAL ARTS and the Diana logo are trademarks of Bloomsbury Publishing Plc

First published in Great Britain 2018

A catalogue record for this book is available from the British Library.

Library of Congress Cataloging-in-Publication Data
Names: Bottazzi, Roberto, author.
Title: Digital architecture beyond computers: fragments of a cultural history of computational design / Roberto Bottazzi.
Description: New York: Bloomsbury Visual Arts, An imprint of Bloomsbury Publishing Plc, 2018. | Includes bibliographical references and index.
Identifiers: LCCN 2017049026 (print) | LCCN 2017049495 (ebook) | ISBN 9781474258166 (ePub) | ISBN 9781474258142 (ePDF) | ISBN 9781474258135 (hardback: alk. paper) | ISBN 9781474258128 (pbk.: alk. paper)
Subjects: LCSH: Architectural design–Data processing. | Architecture–Computer-aided design. | Architectural design–History.
Classification: LCC NA2728 (ebook) | LCC NA2728 .B68 2018 (print) | DDC 720.285–dc23
LC record available at https://lccn.loc.gov/2017049026

ISBN: HB: 978-1-4742-5813-5
ePDF: 978-1-4742-5814-2
eBook: 978-1-4742-5816-6

Typeset by Deanta Global Publishing Services, Chennai, India

To find out more about our authors and books visit www.bloomsbury.com and sign up for our newsletters.

Contents

Preface

Despite digital architecture having carved out an important position within the contemporary discourse, it is perhaps paradoxical that there is still little awareness of the history, fundamental concepts, and techniques behind the use of computers in architectural design. Too often publications concentrate on technical aspects or on future technologies failing to provide a critical account of how computers and architecture developed to converge in the field of digital design. Even more overlooked is the actual medium designers utilize daily to generate their projects. Software is too often considered as just a series of tools; this superficial interpretation misses out on the deeper concepts and ideas nested in it. What aesthetic, spatial, and philosophical concepts have been converging into the tools that digital architects employ daily? What's their history? What kinds of techniques and designs have they given rise to?

The answer to these questions will not be found in technical manuals but in the history of architecture and sometime adjacent disciplines, such as art, science, and philosophy. Digital tools conflate complex ideas and trajectories which can span across several domains and have evolved over many centuries. *Digital Architecture Beyond Computers* sets out to unpack them and trace their origin and permeation into architecture. In introducing the possibilities afforded by the emergent CAD software, W. J. Mitchell (1944–2010) noticed that "the application of computer-aided design in architecture lagged considerably behind applications in engineering. Hostility to the idea amongst architects and ignorance of the potentials of computer technology, perhaps contributed to this (1977, p. 40)." Some twenty years later, it was Greg Lynn's (1999, p. 19) turn to highlight that "because of the stigma and fear of releasing control of design process to software, few architects have attempted to use computers as a schematic, organizing, and generative medium for design." The present situation seems to both contradict and confirm these remarks. If, on the one hand, CAD software has become the media of choice for spatial designers, therefore removing any stigma; on the other, most architects have simply replaced traditional media with new ones, without any substantial effect on their design process or outputs. The conceptual and practical experimentation with computational tools remains a marginal activity in regard to the building industry as a whole. One reason is the

still polarized nature of the debate on digital technologies between detractors and devotees. Both being equally ineffective, albeit in different ways, these factions suffer from the common tendency of approaching the role of digital tools in design too narrowly and, consequently, conjure up "premature metaphysics" of computation (Varenne 2013, p. 97). The former group stubbornly resists acknowledging that digital tools can be used generatively and therefore struggle to grasp the wider, often not even spatial, issues at stake when designing with computers. The latter group, on the other hand, attributes to computers such degree of novelty and internal coherence to self-validate any outcome.

Transformations in the medium of expression of any creative discipline has always had rippling effects on its discourse be it theoretical or practical. For instance, the slow introduction of perspective in the fifteenth and sixteenth centuries elicited the formation of new schools, professions, and reorganization of the building site. Strictly starting from the analysis of the actual design process, the discussion of the case studies eventually branches out to include—whenever relevant—its cascading impact on related fields, such as division of labor, the emergence and propagation of new forms of knowledge or learning, the need for new or different figures, and their impact on public knowledge at large. In fact, the relation between tools for representation and design has always been a fluid one in which the means of expression available at any given time determine the bounds of architectural imagination. As for language, we cannot articulate feelings or ideas for which we have no words; so architects have not been able to build or even imagine forms beyond what is allowed by the tools they employed. The introduction of CAD in the design process has deeply altered the confines of what is possible, but has not altered this basic principle. In Alexandro Zaera-Polo's words "nothing gets built that isn't transposed onto *AutoDesk AutoCAD*." The ambition of this study is to move beyond this sterile position to critically survey the relation between digital means of representation and architectural and urban ideas and forms—whether built or not. The central focus of this research is therefore software, not only understood as an active component of the design, an unavoidable media managing the interaction between designers and designs, but also impacting on the very cognitive structure of design. As such software is always imbued of cultural values—as Lev Manovich (2013) noted—demanding a materialistic, which critically examines its very structure, impact. Whereas critical theory privileges the cultural and social impact of software overlooking its intrinsic qualities and the ideas that actually shaped it, technical manuals offer little scrutiny of the very tools they introduce. This does not necessarily mean that the cultural and social dimensions of digital design will be categorically omitted in the study; rather they will not act as primary engines

for innovation and development in this field. Despite the results achieved by the application of critical theory to many fields, including architecture, when it comes to digital architecture this approach seems structurally unable to grasp the intrinsic qualities, constraints, and issues related to generating spatial ideas with digital devices.

Structure and organization of the book

In introducing his thoughts, Blaise Pascal warned the reader that they should not have accused him of not having said anything original: what was new in his work was the disposition of the material. Likewise, here the reader will recognize names that have been largely discussed in many scholarly works, occasionally they will encounter genuinely new discussions or topics. Regardless, it is the very frame within which these conversations take place to constitute the ultimate novelty of this work: the range of precedents discussed here is brought together for the first time under the agenda of computational and digital design. The formula digital architecture conflates two fields—architecture and computation—whose origins, scopes, and developments are very different from each other and have only recently merged. Modern computers—which only appeared in their modern incarnation during the Second World War—are significantly younger than architecture and built without a precise aim to fulfill—Alan Turing often talked of them as universal machines. By accepting this basic principle, the book expands the history of "digital" architecture beyond the history of computers to highlight how the current generation of digital architects is experimenting with or evolving ideas and techniques that can be traced as far back as the baroque or the Renaissance. The characterization of these episodes is independent of the physical existence of computers at the time, thus implicitly constructing a more cultural history of digital architecture rather than a purely technical one.

Eight chapters, each dissects specific techniques or concepts currently in use in digital architecture and design. As the encounter between architects and computers is opportunistic rather than predetermined, linear and chronological accounts are utilized as little as possible. Instead, the book proposes a sort of archaeology of digital-design processes and methods in which multiple narratives articulated through eight fragments—each corroborated by numerous case studies—through which the discontinuous and fluid trajectory of techniques and ideas can be more carefully articulated. The discussion of each theme contextualizes the use of tools in design: whether it has a generative, representational, operations, or methodological impact. Some tools have limited range of use (e.g., contour), whereas others have impacted several aspects

of design. The discussion of these latter types of tools, such as databases, voxel, and randomness, will be limited to their impact on spatial design and organization.

The eight categories identified, one in each chapter are found in the most popular CAD applications designers employ (morphing, pixel, parametrics, etc.), yet they are not specific to any proprietary piece of software. They form an original and accurate vantage point from which to examine what is at stake when designing with computers. The perimeter of the investigation is software; that is, the interface between the digital—be it computers or other digital devices, such as scanners—and design—its culture, techniques, and communication methods. This is understood in its present configuration, acknowledging that ideas forging them have changed from period to period. The book contains inevitable gaps. This is not only because such "vertical" history privileges the evolution of concepts over an even chronological distribution, but also we abandoned the idea of an encompassing history of the relation between design and computation and only discussed those examples that had a paradigmatic effect on design procedures.

The chapters are arrayed in alphabetical order and it will be left to the reader to conflate, hybridize, evolve, and critically dissect the notions, concepts, episodes, and practice listed in the various chapters. This method will not only mirror the very trajectory through which the practices analyzed came into being in the first place; but will also be a more earnest structure to capture the discontinuous relation between design and computation. As for inventions in other fields, advancements in digital and pre-digital tools often resulted from the more or less smooth fusion of previously separate notions or devices. Out of this process of conflation new affordances emerged; a process which also explains why same episodes, names, or contraptions are recalled in more than one chapter, albeit within a different context.

The book opens with a short overview of the concepts forming the architecture of the modern computer. Besides the key chronological developments, the discussion will also focus on the flexibility and "plasticity" of computation. Computation in fact finds its root in formal logic, a field straddling between sciences and humanities. Its basic architecture underpinning all software architects and designers daily use stems out of a very concise and defined series of the shared procedures. Although for users *Photoshop* and *Grasshopper* may look like very different, if not antithetical, pieces of software, their procedures are not. The chapter on databases—a central component of any piece of software—focuses on the long history of techniques utilized to spatialize data and their, at times, direct impact on architecture. The chapter on networks can

be seen as the extension of the previous one. Whereas databases look at the spatialization of data at the architectural scale, networks are here understood as territorial mechanisms coupling space and information. The growth in scale and complexity of networks is one of the implicit outcomes of this chapter; whereas the long narrative woven by these first two chapters clearly reveals how much the success of embedding information depends on the theoretical framework steering its implementation. Throughout the various cases discussed what emerges is not only an outstanding series of techniques to spatialize data; but also, contrary to common perceptions, how data has always needed a material support to exist, which has often been provided by architecture. "Morphing" discusses a series of techniques to control curves and surfaces which have had a direct impact on the formal repertoire of architects. Part of this conversation overflows into the fourth chapter on the timely theme of parametrics. This is certainly the most popular theme discussed in the book but, possibly for the same reason, the one riddled with all sorts of complexities and misunderstandings. Starting from the great examples of the Roman baroque, the chapter will sketch out a more material, design-driven understanding of parametric modeling. Some of the chapters are not dedicated strictly to computational tools but embrace the composition of the modern computer, which includes digital devices that have little or no computational power. The chapters on pixels and scanners both fit this description, as they chart how technologies of representation ended up impacting design and providing generative concepts. Randomness—the sixth chapter—is unavoidably the most abstract and complex of the whole book. Besides the technical complexity in generating genuine random numbers with computers, it is the computational and philosophical issues which are foregrounded here. Finally, the last chapter discusses notion of voxel tracing both its development and impact on contemporary digital design. The chapter on scanning returns to examine how representational technologies have evolved from mathematical perspective to the laser scanner. Despite being central to many digital procedures, this concept has only recently been explicitly exploited by designers, whereas its historical and theoretical implications have been so far completely overlooked.

Acknowledgments

This book brings together several strands of research that have been carried out over the past fifteen years or so. Many institutions, colleagues, professionals, and students have influenced my views for which I am very thankful. I am particularly thankful to Frédéric Migayrou not only for providing the afterword to the book, but also for giving me the opportunity to develop my research and for sharing his time and immense knowledge with me. At The Bartlett, UCL—where I currently teach—I would also like to thank Marjan Colletti, Marcos Cruz, Mario Carpo, Andrew Porter, Mark Smouth, Bob Sheil, Dr. Tony Freeth, and Camilla Wright. At the University of Westminster—where I also work—I am particularly grateful to Lindsay Bremner for broadening the theoretical territory within which to discuss the role of computation in design, Harry Charrington, Richard Difford, Pete Silver, and Will McLean. During the ten years spent at the Royal College of Art, Nigel Coates was not only first to believe in my research, but also communicated me a great passion for writing and publications in general. I am also grateful to Susannah Hagan—whose *Digitalia* injected a design-driven angle to the work—Clive Sall. Amongst the many outstanding projects I followed there, Christopher Green's had an impact on my conception of digital design. Parallel strands of research were developed whilst at the Politecnico of Milan where I would like to thank Antonella Contin, Raffaele Pe, Pierfranco Galliani, and Alessandro Rocca. The section on experimental work developed in Italy in the 1960's and 1970's is largely based on the generosity of Leornardo Mosso and Laura Castagno who gave me the opportunity to analyse their work, Guido Incerti, and Concetta Collura. The research on the use of digital scanners on architecture was also developed through conversations with ScanLab and Andrew Saunders. Over the years some key encounters have changed my views of architecture which have eventually opened up new avenues for research which have converged in this book. These are: Oliver Lang, and Raoul Bunschoten.

A special thank you to my teaching partner, Kostas Grigoriadis for his insights, commitment, and help. I am also grateful to Bloomsbury Academic for the opportunity they provided me with; particularly, James Thompson—my editor—who supported this project and nurtured it with is comments, Frances Arnold, Claire Constable, Monica Sukumar, and Sophie Tann.

Finally, I would like to thank my parents for their continuous support. My deepest gratitude goes to my wife Stefania and my children Aldo, and Emilia who have not only endured the hectic lifestyle which accompanied the preparation of the book, but have also supported and encouraged me in every which way possible. Stefania followed the entire process of this book providing invaluable knowledge and critical insights at all levels: from the intellectual rationale framing the work to the detailed feedback on the actual manuscript. Without their unconditioned love and help all this book would not have existed. It is to them that this book is dedicated to.

Illustrations

Introduction: Designing with computers

A machine is not a neutral element, it has got its history, logic, an organising view of phenomena

Giuseppe Longo (2009)

Before venturing into the more detailed conversations on the role of digital tools in the design of architecture and urban plans, it is worth laying out a series of the key definitions and historical steps which have marked the evolution and culture of computation. Whereas each chapter will discuss specific elements of computer-aided design (CAD) software, here the focus is on the more general elements of computation as more abstract and philosophical notions. Built on formal logic, computers are unavoidably abstracting and encoding their inputs; whatever media or operation is eventually transformed into strings of discrete 0s and 1s. What is covered in this short chapter is in no way exhaustive (the essential bibliography at the end of the chapter provides a starting point for more specific studies) but clarifies some of the fundamental issues of computation which shall accompany the reader throughout all chapters.

First, computers are logical machines. We do not refer to a supposed artificial intelligence computers might have, but rather, literally, to the special branch of mathematics that some attribute to Plato's *Sofist* (approx. 360 BC), which concerns itself with the applications of principles of formal logic to mathematical problems. Whereas formal logic studies the "structure" of thinking, its coupling to mathematics allows to broadly express statements pertaining to natural languages through algebraic notations, therefore coupling two apparently distant disciplines: that of algebra and—what we now call—semiotics. It is this century-long endeavor to create an "algebra of ideas" that has eventually conflated into the modern computer, compressing a wealth of philosophical and practical ideas spanning over many centuries. The common formal logic from which digital computation stems also accounts for the "plasticity" of software: beyond the various interfaces users interact with, the fundamental operations

performed by any software eventually involve manipulation of binary code consisting of two digits: 0 and 1. This also explains why an increasing number of software can perform similar tasks: *Photoshop* and visual scripting language *Grasshopper*, for instance, allow to manipulate similar media objects, such as geometries and movies.

What's a computer, anyway? It is easier to see how the etymology of the word derived from the act of computing, of crunching calculations; however, whenever we buy a computer we actually purchase a series of devices only some of which actually compute. Computers in fact also include input devices—keyboard, mouse, etc.—and output ones—monitor, printer, etc.—allowing us to interact with the actual computing unit. Computation is therefore an action which is not exclusive to computers; likewise, the word "digital" does not solely pertain to the domains of computer, as it derives from digits, discreet numerical quantities (such as those of the fingers of our hands).

Perhaps unsurprisingly given these initial definitions, modern computation is a rather old project whose foundation can be identified with the groundbreaking work of Gottfried Leibniz in the seventeenth century. In Herman Goldstine's words (1980, p. 9), Leibniz's contribution can be summarized in four points whose power still resonate with the work of digital designers today: "His initiation of the field of formal logic; his construction of a digital machine; his understanding of the inhuman quality of calculation and the desirability as well as the capability of automating this task; and, lastly, his very pregnant idea that the machine could be used for testing hypothesis." It is this very last point to both reveal the importance of Leibniz's thinking and set a richer context for digital design: this field is still somehow stigmatized for its "impersonal," rigid rules stifling the design process, whereas anybody fairly fluent in digital design would know that the opposite is also true. Computers' ability to take care of the "inhuman quality of calculations" frees up conceptual space for the elaboration of alternative scenarios. As for testing hypotheses, it implies an experimental, open, iterative relation between designer and computer aiming at fostering innovation.

Before surveying some of the steps toward the construction of modern computers, it is important to clarify some of the key concepts we will repeatedly utilize throughout the book.

Analogical and digital computing

Prior to the invention of modern computers in the first part of the twentieth century, the most advance calculating machines utilized analogous or continuous computation. Analogue forms of computation were based on

continuous phenomena. All empirical phenomena are analogical; they always occur within a continuum. For instance, time flows uninterrupted regardless of the type of mechanism we are using to measure it. Analogue computers execute calculations adopting continuous physical elements whose physical properties are measured—for example, the length of rods is recorded or different current voltage. As Goldstine (1980, p. 40) reminds us, analogue computing goes hand in hand with nineteenth-century mathematics: the developments in the field of mathematics required new types of machines—precisely, analogue machines—in order to compute the set of equations describing a certain physical phenomenon. "The designer of an analogue device decides what operations he wishes to perform and then seeks a physical apparatus whose laws of operation are analogous to those he wishes to carry out." The slide ruler is an example of analogue computing in which logarithms are calculated by sliding two markers along a graded piece of wood, effectively measuring their position along a graded edge. The two markers physically represent established mathematical properties of logarithms stating that the logarithm of a product of two numbers is the sum of the logarithms of the two numbers. Through these examples it is possible to discern how computational machines are always an embodiment of theory, and they are not natural but designed artifacts, informed by theoretical preoccupations as well as physical limitations.

Modern computers, on the other hand, are digital machines; they operate with digits combined according to algebraic and logical rules. They do not operate with continuous quantities—like their analogue counterparts—but rather discrete ones which capture through numbers what would otherwise be a continuous experience. The invention of the Western alphabet could very well mark the introduction of the first discrete system: whereas when we speak the modulation of sounds is continuous, the alphabet dissects it into a number of defined symbols—letters. The abacus, Leibniz's calculating machine, and Charles Babbage's (1791–1871) *Analytical Engine* (proposed in 1837) are examples of discrete computing machines in which basic calculations such as additions and subtraction are executed and the results carried over to complete other operations. In the case of Leibniz's wheel numerical quantities are engrained on metal cogs which click to position to return the final desired results. Quantities are finite and discreet, no longer continuous. Modern computers always discretize; they reduce continuity to the binary logic of 0s and 1s creating a problematic conceptual and, at times, practical gap between the natural and the artificial. These problems are at the center of studies on how computers operate as ontological and representational machines; though these conversations affect all areas of computations, this book will particularly concentrate on its impact on

design, especially in the use of computer simulations and parametric modelers to design architecture.

The elegance of binary code

As we mentioned, all data input in or output by computers are formatted in binary code. The elegance of this system brings together several disciplines and knowledge which have matured over many centuries. We know that the invention of binary code greatly precedes its introduction in Leibniz's work. The German philosopher's was, however, motivated by a different desire; that of conceiving the shortest, in a way the most economic numerical system able to describe and return the largest number of combinations. Leibniz in fact saw binary numbers as the starting point of a much bigger endeavor: that of expressing philosophical thoughts and even natural language statements through algebraic expressions. Leibniz did make several attempts to both define such a system and test its applications—for instance, to resolve legal disputes—without much success: this would fundamentally remain a dream—to borrow Martin Davis' expression—that would be quickly forgotten after his death.

Uninterested in Leibniz's philosophical ambitions, French textile worker Basile Bouchon developed a system of perforated cards in 1725 to control the weaving patterns of mechanical looms, which was shortly after improved by Jean-Baptiste Falcon. Once the cards were fed through the machine, the loom would automatically alternate the combination of threads to obtain a desired pattern. Binary logic would find an ideal partner in the material logic of perforated cards as the unambiguous logic of either holed or plain cells mirrored that of 0s and 1s. Despite the invention and rapid diffusion of microprocessors, mainframe computers still used punch cards as material support for software instructions.

Whether aware of Leibniz's work or not, binary numbers were also the decisive ingredient in British mathematician George Boole's (1815–64) work whose logic basically marked the modern foundation of this discipline and indelibly shaped how computers work. Boole's (1852, p. 11) own words well capture the importance of his work: "The design of the following treatise is to investigate the fundamental laws of those operations of the mind by which reasoning is performed: to give expression to them in the symbolical language of Calculus, and upon this foundation to establish the science of Logic and instruct its method; . . . and finally to collect from the various elements of truth brought to view in the course of the inquiries some probable intimations concerning the nature and constitution of the human mind." Boole's invention consisted of

employing algebraic notation to express logical statements; such connection was just a brilliant example of scientific thinking but also provided a clear and coherent bridge between algebra and natural languages. Using the four arithmetical operations, Boole could translate statements in natural language. For instance, the * symbol describe a "both" condition: the group of all red cars could be expressed as, for instance, $x = y * z$ or $x = yz$; in which y describes the group of "red objects," whereas z denotes that of "cars." The + symbol described and/or conditions, from which it was possible to quickly infer how the symbol could also be utilized. The famous exception to the algebraic notation—which had already been anticipated by Leibniz—was the expression $x^2 = xx = x$, as adding a group to itself would not produce anything different or new. Boole (1852, pp. 47–48) introduced binary numeration as "the symbol 0 represents Nothing," whereas the symbol 1 represents "'the Universe' since this is the only class in which are found *all* the individuals that exist in *any* class. Hence the respective interpretations of the symbol 0 and 1 in the system of Logic and *Nothing* and *Universe*" (italics in the original).

From the point of view of computation, Boole's logic basically allowed to program a computing machine: it supplied a syntax to correctly turn instructions—linguistics—into machine commands—numbers. It is therefore not a coincidence that further work in this area—particularly by Gottlob Frege (1848–1925)—also marked the beginning of modern studies on semiotics. However, the full realization of this potential would only occur in 1910–13 when Bertrand Russell (1872–1970) and Alfred North Whitehead (1861–1947) would publish their *Principia Mathematica* taking up Boole's logic (another failed dream, in some respect).

The nineteenth century was also characterized by progresses made in the development of electricity. Electric circuits would eventually be employed to control machines and become the "engine" of the modern computer. Switches in electrical circuits also only have two positions: they are either open or close. Peirce first intuited that binary code could have been the ideal language to control the position of the switches. Its application to computation would, however, only occur in the 1930s when Claude Shannon's essential work on information—also discussed in the chapter on randomness—would relate formal logic (by now, programming), electrical circuitry, and information transmission under the unifying language of binary numbers. After Shannon's work it was possible to compute with a modern computer: that is, to determine a set of logical steps, translate them into a programming language engendered with both semantic and syntactic characteristics, which could instruct the electric apparatus of the computer.

This short foray into the evolution of basic programming language for computers shows how computers came to exploit disparate notions which eventually converged; since the Jacquard's loom, computation has been consisting of a hardware (computing mechanism) and a software (set of instructions), which will be briefly discussed here.

Data and information

Information is one of the key words of the twentieth century. Its relevance has increased exponentially not only through the proliferation of new media, but also through the parallel interest expressed by more established disciplines, such as philosophy and ethics. The invention of the modern computer surely played a significant part in growing its popularity: in fact, the computer essentially performs nothing but manipulations of stored information. The wealth of knowledge on the subject has not always been beneficial, as several definitions of the same words emerged responding to the very contexts in which they were analyzed. Information and data have been defined in semantic, statistical terms often presenting contrasting definitions.

Rather than attempting to reconcile these differences, we concentrate on the very material nature of the modern computer and on its intrinsic qualities and limitations. Computational information is purely a quantitative phenomenon, unrelated to qualitative, sematic concerns: it can claim no meaning, and even less truthfulness. To understand the nature and properties of digital data and information, we have to cast a larger net of categories over the subject. Contrary to the superficial notion that data is abstract, immaterial entity, computers first and foremost are material constructs: data stored in computers exist as the combinations of physical properties. Most often these are alternate voltages switching between two currents corresponding to binary numeration. Binary digits—better known as bits—are the building blocks of digital data. There are sequences of 0s and 1s, without any precise meaning: they could be characters in a text, songs, 3d models, etc. Groups of bits form patterns used as codes. Structured and coded strings of bits are finally defined as data, whereas information is generally defined as data in context. In this "epistemological chain" of digital data, information marks the threshold in which the material properties are stored in the hardware can designed. This specific act is performed through algorithms which allow information to be "interpreted, manipulated, and filled with meaning" (Ross 1968, p. 11; quoted in Cardodo Llach 2012, p. 42). Strictly speaking, digital media only deal with information as this is the deepest editable layer of content in the computer; for this reason we

have a specific field of studies dedicated to information—i.e., informatics—but not to data.

Brief history of computers

The modern computers emerged out of the millennia-long development of artificial apparatuses to assist human calculation. Its origin is found in the development of calculating machines first manually operated and then based on activating mechanical parts. Among the most ancient computing devices we can count the Antikythera mechanism—perhaps the first ever—an analogue computing orrery discovered on the homonymous shipwreck in Greece. Made of about thirty interconnected bronze cogs, this device could have been made between 150 and 100 BC and used for astronomical calculations. The abacus, a calculating machine based on discrete quantities, emerged much later, around 1200 in China. The emergence of mechanical calculating machines is generally understood to coincide with Blaise Pascal's (1623–62) device built in 1642—at the age of twenty—for his father. The machine, based on a series of rotating wheels, could solve additions and subtractions. Not long after, in 1673 Leibniz completed his version of a similar type of machine—often referred to as "Leibniz wheel" because of its operating principle—which extended its functions to all four basic mathematical operations.

Falcon's perforated cards were further developed by Joseph Marie Jacquard (1752–1834), who connected them to a weaving loom neatly separating the set of instructions to compute—marking the birth of the notion of software—the

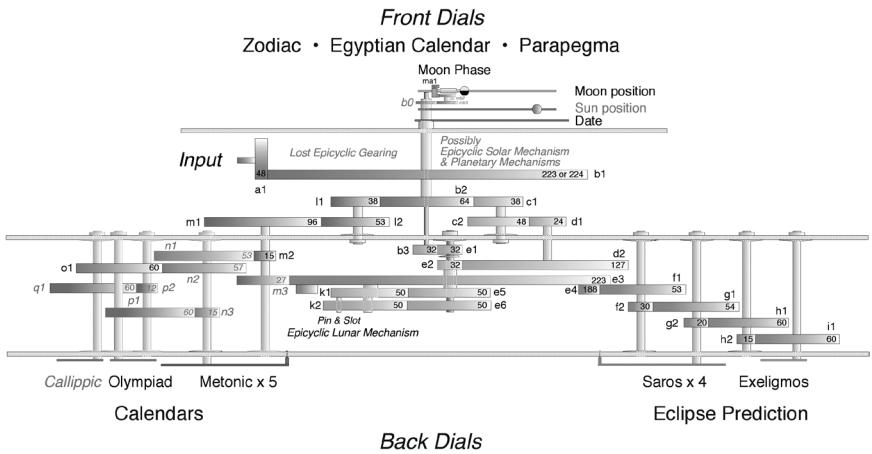


Figure 0.1 Antikythera Mechanism. Diagram by Dr. Tony Freet, UCL. Courtesy of the author.

device physically computing it—from hardware. This division still in use was central not only to the application of computing technologies to everyday tasks but also to the emergence of information as a separate field in computational studies. It is interesting to point out the impressive penetration that this machine had, once again demonstrating that computation is not a recent phenomenon: in 1812 there were 11,000 Jacquard looms in use in France.[1]

The principles of the Jacquard loom were also at the basis of Charles Babbage's *Difference Engine* (1843). Operated by punch cards, Babbage's machine could store results of temporary calculations in the machine's memory and compute polynomials up to the sixth degree. However, the *Difference Engine* soon evolved into the *Analytical Engine* which Babbage worked on for the rest of his life without ever terminating the construction of what can be considered the first computer. Its architecture was in principle like that of the Harvard Mark I built by IBM at the end of the Second World War. The working logic of this machine consisted of coupling two distinct parts, both fed by perforated cards: the store, which computed the logical steps to be operated upon the variables, whereas the mill stored all the quantities on which to perform the operations contained in the store. This not only meant that the same operations could be applied to different variables, but also marked the first clear distinction between computer programs—in the form of algebraic scripts—and information. This section would not be complete without mentioning Augusta Ada Byron (1815–52)—later the Countess of Lovelace—whose extensive descriptions of the *Analytical Engine* not only made up for the absence of a finished product, but also, and more importantly, fully grasped the implication of computation: its abstract qualities which implied the exploitation of combinatorial logic and its application to different type of problems.

The year 1890 was also an important year in the development of computation, as calculating machines were utilized for the U.S census. This not only marked the first "out-of-the-lab" use of computers but also the central position of the National Bureau of Standards, an institution which would play a pivotal part in the development of computers throughout the twentieth century: as we will see later, the Bureau will also be responsible for the invention of the first digital scanners and pattern recognition software. The technology utilized was still that of perforated cards, which neatly suited the need to profile every American citizen: the organization in rows and columns matched the various characteristics the census aimed to map. The year 1890 marks not only an important step in our short history, but also the powerful alignment of computers and bureaucracies through quantitative analysis.

Whereas the computing machines developed between 1850 and the end of the Second World War were all analogue devices, the ENIAC (Electronic Numerical Integrator and Calculator), completed on February 15, 1946, emerged as the first electronic, general-purpose computer. Contrary to Vannevar Bush's machines developed from the 1920s until 1942, the ENIAC was digital and already built on the architecture of modern computers that we still use. This iconic machine was very different from the image of digital devices we are accustomed to: it weighed 27 tons covering a surface of nearly 170 square meters. It consisted of 17,468 vacuum tubes—among other parts—and was assembled through about 5,000,000 hand-soldered joints needing an astonishing 175 kilowatts to function. It nevertheless brought together the various, overlapping strands of development that had been slowly converging since the seventeenth century, and, at the same time, paved the way for the rapid diffusion of computation in all aspects of society.

The final general configuration of modern computers was eventually designed by John von Neumann (1903–57) whose homonymous architecture would devise the fundamental structure of the modern computer as an arithmetic/logic unit—processing information; a memory unit—later referred to as random-access memory (RAM); and input and output units (von Neumann 1945). The idea of dedicating separate computational units to the set of instructions contained in the software from the data upon which they were operated allowed the machine to operate much more smoothly and rapidly, a feature we still take advantage of.

The 1970s finally saw the last—for now—turn in the history of computers with the emergence of the personal computer and the microprocessor. Computers were no longer solely identified with colossal machines that required dedicated spaces, but rather could be used at home and tinkered with in your own garage. This transformation eventually made processing power no longer "static" but rather portable: today roughly 75 percent of the microprocessors manufactured are installed not on desktop computer but on portable machines like laptop, embedding computation into the very fabric of cities and our daily life.

Brief history of CAD

It was the development of specialized pieces of software that marked the advent of CAD tools. The invention of CAD should be seen as one of the products of the conversion of the military technologies developed during the Second World War to commercial uses as needed by the US government in order to capitalize on the massive investments made. This decision had a profound effect on postwar