Executing Practices

Helen Pritchard, Eric Snodgrass, Magdalena Tyżlik-Carver

Towards the end of a keynote address on "Theory and Practice" presented in 1989 at the 11th World Computer Congress, the well-known computer scientist and mathematician Donald Knuth suggests a challenge to his audience.

> *Make a thorough analysis of everything your computer does during one second of computation. The computer will execute several hundred thousand instructions during that second; I'd like you to study them all.* (Knuth [1989] 1991, 12–13)

There is an expectation that comes from a technical understanding of execution that it is a straightforward running of a task. For instance, in computing, execution is often associated specifically with the fetch–decode–execute instruction cycle, during which a computer's central processing unit (CPU) retrieves instructions from its memory, determines what actions the instructions dictate and proceeds to carry out those actions. But of course the instruction cycle does not encompass execution's impact and embeddedness in the world, and it is this that contributors to this book elaborate and expand upon critically. As Knuth notes, "[e]ven when the machine's instructions are known, there will be problems" (13).

Contained in every "blip" of execution is a range of technical and cultural issues to be addressed, with one operational experience of executing practices opening onto another (Fuller 2003).[1] *Executing Practices* brings attention to what Isabelle Stengers (2005) describes as the particular demands of practices that propel execution. Practices are parsed as processes by which execution stabilises and takes hold in the world (Stengers, in Gabrys 2016, 9). Rather than considering the stability of execution as the norm, which we might approach with dystopic or paranoid dread, the authors in the book engage with and make interventions on the problems of execution.

Executing Practices alerts us that access to instructions that drive execution is only one account, and even then, our understanding of execution might always remain partial and speculative. If we approach Knuth's challenge through an engagement with practices, it becomes apparent that processes of computation have particular obligations that infringe upon those who practise or are affected by it. Through geographic, temporal and material specificity the chapters attend both to the practices of execution and their differing research practices. The focus is on complexities inherent to different forms of execution, while also recognising an understanding of execution as a performance of step-by-step instructions. The outcome of this is a

collection of research practices that intervene in executing processes at differing points and locations to engage with the most important aspect of Knuth's challenge—*the problems* of execution.

"Uwaga … Start!!": Experiences of Execution

The practices of the women who devised and implemented the programming for ENIAC (Electronic Numerical Integrator And Computer) in the 1940s might offer a useful orientation when addressing Knuth's challenge. If we consider one second of computing in this example it becomes clear that it is not just algorithmic calculations that have to be attended to but also women setting values, connecting switches and wiring cables and plugs between different parts of the machine (what is now referred to as "direct programming"). At a time when there was no computer language and no operating system as such, "the women had to figure out what a computer was, how to interface with it, and then break down a complicated mathematical problem into very small steps that the ENIAC could then perform" (Kathy Kleinman, in Sheppard 2013; see also Chun 2004, Hayles 2005, Balsamo 1996). The working system which supported their invention of coding, with its various hierarchies and divisions of labour, was described by Jean Jennings, one of the ENIAC operators, in the following way:

> *Betty and I were the workhorses, finishers, tying up all the loose ends. Kay was often more creative, suggesting clever ways to reduce total size of the program. Marlyn and Ruth agreed to generate a test trajectory, calculating it exactly the way the ENIAC was to do it so we could check the detailed steps once it was on the ENIAC. We spent a lot of time working on programming notation so we could keep track of the timing of program pulses and digital operations. The ENIAC was a parallel machine, so the programmer had to keep track of everything, whether interdependent or independent.*
> (Jennings, in Fritz 1996, 20)

Computing here, as well as being a physical execution of calculations that require wiring by hand, is also a task of military labour which is divided according to skills that demand an intimate understanding of the machine and processes required to run it. Situating ENIAC's practices is also important. ENIAC was initially sponsored by the US military as a general-purpose electronic computer for calculating artillery-firing tables (the settings used for different weapons under varied conditions for target accuracy), and later for other tasks such as numerical weather prediction and the working out of implosion problems relating to the ongoing development of the hydrogen bomb. In this account of computational practices, the problems of execution are historically situated and entangled with the contingent

forces of machines, bodies, institutions, military labour practices and geopolitics, rather than simply a set of instructions that are outside of life.

Another example that highlights differing experiences of execution is the idiosyncratic coding practice of Radiokomputer that developed in Poland in the late 1980s. Radiokomputer illustrates the distributed relations to be taken into account when thinking about execution and how execution might be experienced. Radiokomputer[2] was a radio programme broadcast on Polish National Radio between 1986 and the early 1990s, transmitting via shortwave frequencies computer programs and games for early home computers such as Atari, ZX-Spectrum and Commodore 64. A similar distribution of music via radio was commonly practiced for most of the 1980s, when radio presenters would broadcast boths sides of vinyl LPs delivered or smuggled to Poland from West Europe. Political restrictions on culture and commerce at the time influenced and generated particular ways of sharing foreign pop culture. It is not surprising that this model was also used for distributing computer programs, which were radiocast for the listeners to record onto a cassette tape. At 4pm on Fridays after a brief introduction, the radio presenter would announce the transmission with a warning to listeners: "Uwaga … Start!!" which would mark a moment to press the record button on a tape recorder, after which a nationwide broadcast of noise would follow. As one of the programme listeners recalls, Spectrum sounds would differ from Atari, and Commodore would also sound recognisably different.[3] Unfortunately, this cacophony of sounds would not always deliver, as any interference in radio waves could corrupt the program. According to computer users at the time, there was an estimated 70% success rate for this form of program recording, with Atari being the most amenable to this method and Spectrum being least open to it. To aid the process, the radio presenter wrote an article advising the best recording practice, which was then published in *Bajtek*, a monthly journal dedicated to computers and related technologies (for more details, see Jordan 1986). The articles included step-by-step instructions, with information about what hardware to use (Polish cassettes produced by Stilon were not recommended because of the low level of iron necessary for better quality of recording) and how to set up for best results (including the advice to turn off all unnecessary electrical devices in the house, such as washing machines, hoovers, etc.).

Practices such as these highlight what are perhaps less familiar experiences of computing. In Radiokomputer, the socio-political situation and lack of copyright laws regarding software in Poland at the time generated a practice of national broadcast radio for free transmission of code. On Friday afternoons, as long as the radio was

tuned to the right station, it was possible to listen to code and hear its crackling noises while attempting to record it so that it could be executed again as a game. This example is another instance of an executing practice which, together with the example of ENIAC, points to localised and physical experiences of code. A multiplicity of relations are highlighted in such executions, which, as well as including hardware and software, are also dependent on laws, cables, the electromagnetic spectrum, minerals, histories, gender relations, economies and so on. Issues of maintenance and instantaneous debugging are at the very centre of this form of code writing, inscribing computational ecologies as unexpected systems that are as temporary as they are concrete in the moment of their execution. And so investigations of execution pay attention to which stories of execution we choose to tell and which are forgotten in the history of software.

Where should we conclude this readily sprawling task of practicing and working through execution as inquiry? This is a key question, and as contributions to this volume suggest, whilst accounts might reveal the terminal character of computation, there is no end to such investigations. For instance, Knuth's challenge could be considered to be a practical study in which one remains within the physical confines of the machine itself: a world of circuitry-registers, operational codes, scan codes, glyph selections, screen renderings, non-keyboard inputs and the like. In addition, this "your computer" is itself connected to the distributed services of the Internet, subject to and executing within "local" and "global" experiences of packet switching, resolutions of internet protocols, scripts, multiple caches and loads, and so on. And what then of the busy electrons and swerving atoms charging the "bare metal" and flowing onwards within greater infrastructures of electricity, optical fibre, manufacturing and so on? And what of the different collective entities and bodies that necessarily act as transducers for such energies? Knuth's problem opens further still and in come the uninvited guests of perspectivalism, political economy and the general meshed nature of the world. In the meantime, the complexity and amount of actions performed by a typical computer have increased exponentially. As one commenter on a *Hacker News* thread replied to the question of what happens when you type Google.com into your browser and press enter? "Somebody also needs to talk about what's happening in the CPUs, with 3 billion or so instructions per CPU core every second, all devoted to looking up a cat video for you. When you play a cat video, more computation occurs than was done in the history of the world prior to 1940" (Animats 2015).

Beyond standard attempts aimed at unpacking discrete instances of execution—typically carried out with the intention of optimising

the executing processes involved — the notion of tracking execution and its many shifting parts over a particular instance of time has produced a variety of responses on the part of practitioners and artists. In *Diff in June* (2013), artist Martin Howse uses a small bit of custom script to track whenever a bit of data is changed between one day and the next within the file system of an IBM x60 machine. Running the script results in a 1,673 page transcript that creates a narrative of "a day in the life of a personal computer written by itself in its own language, as a sort of private log or intimate diary focused on every single change to the data on its hard disk" (Howse 2013). In this book, David Gauthier's contribution *Loading… 800% Slower* enacts a method of détournement that willfully slows down the bitrate of an internet connection, making audible the many "timely designed assaults" of the invisible scripts involved in composing a particular web page. Magdalena Tyżlik-Carver and Andrew Prior assemble code, interface, texts and sound in a *Ghost Factory* experiment that makes recursivity available to participating bodies, whether human or not. Elsewhere, the excessive character of execution as a form of eroticism is hacked by *Marcelle*, a pair of white cotton briefs equipped with vibrators that respond to surrounding WiFi networks. An intervention by Marie Louise Søndergaard which, as further discussed in her joint article with Kasper Hedegård Schiølin, functions as a conceptual tool that posits eroticism as "an inherent aspect of computational culture and history". Meanwhile, Olle Essvik investigates execution as a practice of bookbinding that incorporates book-end papers bought at an auction in Sweden. In the process he explores random noise generation and "chance executions" by referencing situated material histories whose traces are found on the purchased papers and then performed in the making of the book. Such methods and their often performative modes of "parasitic rendering" (Gauthier) bring to the fore inflecting and productive relations of even the most minor executing procedures.

The contributors to this collection account for both the practical specificities of computing and a range of matters both very close to and also, seemingly, very far from the machine itself. In particular, the book presents why, and in which ways thinking through a notion of execution can be useful. Each piece in the book provides its own response. Some work towards defining a particular mode or process of execution, and others use execution as a concept through which to study a variety of issues and their relations to one another. As writers such as Karen Barad (2007) make clear, the path towards answering a question such as Knuth's will say much about the ontologies, epistemologies and various ensembles of objects and entities brought together in answering it. It is because of this complex character of computation that questions such as Knuth's are commonly brought

up during job interviews in computing and related fields. Ask a Java programmer what they understand execution to mean and you will likely get a rather different answer to that of someone involved with physical computing or a researcher working within the fields of queer theory or software studies. Such accounts of execution point to complex relations that are highlighted in practices, opening up an understanding of execution to its different experiences.

While each contribution in the book covers differing experiences of execution, we will highlight a few tentative themes shared by many of the chapters. The intention is not to categorise the contributions or map out a definitive set of themes, but rather to give a sense of some of the directions which working through a notion of execution takes us.

Executing Temporalities

Today it is no longer a couple of hundred thousand instructions executing per second (as Knuth suggests in 1989), but rather an accelerating number of potential instructions at any one time. One practical way in which to deal with Knuth's suggestion on the typically much faster machines of the present would be to cut a single second into a more manageable unit of time: perhaps a nanosecond (one billionth of one second), the time it typically takes to execute one machine cycle on a 1 GHz microprocessor. If we take computational time to be linear — in the way that Knuth's challenge might suggest — the focus is on that moment in read-write culture where the computer program "does what it says". Execution is often considered as a culminating step in writing a program, yet at the same time it is but a split moment in computer time: a second that is instantaneous with another second, and another and so on.

As Winnie Soon's and Brian House's essays in this book both argue, computation depends upon increasingly brief and strictly maintained micro-temporalities, in which the maintaining of a consistency in signal processing is essential for the establishing of clock cycles, both in local and more global instances of computation. Thus, as House's essay explains, Google Spanner's "TrueTime" Application Programming Interface (API) is a practical method for synchronising the executing uncertainties of individual computer time in relation to the various needs of Google's globally networked systems. Nevertheless, like the many timekeeping strategies before it, in the process of doing so, Google Spanner inevitably has a direct role in establishing various forms of "micro-experiences" for the many users that come within its sweep (House). Soon traces this micro-temporality of computers and the network back from the planetary scale to the rather more mundane instance of a "throbber", those pulsating images of spinning wheels that for Internet users

signify a time of waiting for a stream of information to resolve itself. As Soon explains: "a throbber icon acts as an interface between computational processes and visual communication", thus echoing Wendy Chun's well-known statement that software creates an invisible system of visibility by obfuscating certain structures while revealing others (2004, 27). In this sense, the throbber can be understood as an obfuscation of the necessarily discontinuous executing processes of discrete computing, replacing the asynchronous and uncertain clockworks of these tasks with an intentionally smoothed-out visual presentation of the network. Thus a throbber, like Google Spanner's TrueTime, is itself yet another cultural and computational practice that plays a role in "constantly rendering the pervasive and networked conditions of the *now*" (Soon).

In his preface to this book, Yuk Hui notes that "[e]xecution is always teleological because to execute means to carry out something which is already anticipated before the action". Any particular *telos* can be reached according to different methods, each with their own temporalities and often isometric worldviews. Hui traces the way in which a largely linear temporality with predefined sequential procedures and relative logical certainty—such as one finds in eighteenth and nineteenth century forms of mechanisation—represents both an intuitive and simple method of application in executing procedures. At the same time, such perspectives can be seen to readily coevolve with the material and economic conditions of the time in question. The eventual arrival of general-purpose electronic computing machines in the twentieth century sees an explosion of linearity into non-linear recursive cycles of execution. In the process, this introduces different potential rhythms of mechanisation and related paradigms for understanding the world; with the implications of automation and the steady rise of platform capitalism posing particularly urgent questions for enquiry.

In his separate article contribution, Gauthier interrogates misplaced notions of executions as apodictic commands to be followed. In opposition to this sense of command as control, he highlights practices of debugging as illustrative of the continual and unpredictable itineration of signs and signals working themselves through the architectures of any given machine at a given time. The term execution and the way in which it emphasises a sense of a decisive moment can risk a similar emphasis on foreclosure. In contrast, the equally common terminology of running a program has the effect of shifting the focus to a sense of the durational aspects of live execution (runtime) and the ongoing, necessary processes of maintenance involved in executing systems—a topic which Linda Hilfling Ritasdatter's article explores. Her ethnographic investigation in Chennai, South India into the Y2K

problem at the turn of the millennium gives a poignant example that links maintenance to a number of problems, including those of computation and its economic conditioning as well as particular colonial and other historical trajectories.

Executing Ecologies

As contributions to this book show, execution is not simply a clean delivery of a task. Command and control is never absolute. This is not to say that a program does not do what it says. Rather, the authors focus on what execution is, how it operates and what might be obscured in the process. The history of computing is one in which computation, in its actual execution and spreading into domains of all kinds, inevitably grows *wild.* As media theorist Friedrich Kittler aptly states,

> *David Hilbert's dreamlike program to clear out the opacity of everyday language once and for all through formalization is undone not only at the clear, axiomatic level of Gödel or Turing, but already by the empiricism of engineers. Codes with compatibility problems begin to grow wild and to adopt the same opacity of everyday languages that have made people their subjects for thousands of years.* (Kittler 1997, 167)

Knuth himself, in an aside during the same keynote, hints at this unruly expansiveness of computing in the world. He refers to a recent experiment carried out by researchers looking to identify and count each tree in a tropical forest. By Knuth's reckoning, the process of counting 250,000 trees in the arboreal survey was roughly equivalent to the number of instructions in a second of computing at the time (Knuth 1991, 13). What, one may ask, is the point of this seemingly off-handed comparison, in which Knuth sees fit to even include detailed photocopied samples from the article on the tree survey in his slides for the keynote presentation? A response suggested by this book would be that enumeration, as a theory and practice lying at the core of computing, puts into motion further modes of counting and calculative execution. Francisco Gallardo and Audrey Samson give the example of Charles Darwin's work on evolutionary deviation from the norm, highlighting how, with the gradual maturation of statistics, theory becomes fully provable as a "thing that holds" (Alain Desrosière, cited in Gallardo and Samson); in other words, as a theory that becomes a fully executable practice. To parcel out the mathematical or the technical from the many other relations that Gallardo and Samson point to, is to miss one of the key qualities and emphases of execution as the direct experimentation with various materially directed affordances and relationalities. This becomes that, and along the way, becomes something entirely else, with each execution posing further correlations, problems and interpretations to be addressed (Snodgrass).

As Jennifer Gabrys notes in the collection's afterword, execution "is a process and condition that might unfurl through code, but also overspills the edges of code". Such intensifications of computation into the lived, everyday experience and its situated applications introduce ecologies that bring other figures of execution that operate outside of a relatively stabilised domain of computation. Contributions in this book include sound, image, user practices, popular culture and shrimping alongside computation. In these instances, execution is often treated as a bio-geo-political process that engages complex terrains. The skins of mammals become sites for pincer-like executions by tick or computer (Snodgrass). Transgenic fish and microbes become organisms where execution is increasingly instantiated (in both a metaphysical and computational sense) by the extension of computation into biotic subjects (Pritchard). Brown shrimp (*Crangon crangon*), fishing trawlers and mechanised modes of automation exist within critical territories of extinction (Gallardo and Samson). In other articulations of natureculture, content curating functions through practices of linking, liking, reposting, RSS feeds or even contouring, while making users' bodies operational for the purposes of big data (Tyżlik-Carver). Hard-coded forms of self-representations such as one finds in the example of emoji character sets are governed by Unicode protocols and the dominant corporate interests of the present (Pierrot, Roscam Abbing and Snelting). Bodies of many kinds become malleable materials that introduce both flexibility, resistance and often unruly factors of contingency into execution.

Executing Politics

Computing, as an endeavour which emerges out of concerted efforts at command and control, has demonstrated a propensity for furthering the range of executable tasks towards which it can be applied. We find ourselves in an era of an Internet of Things in which computation insinuates itself into objects such as fridges (Gabrys), deadly executions by remotely controlled and autonomous drones (Schuppli) and executions that take place in toxic and polluted landscapes (Pritchard). This increasingly wide range of executing things and practices has the effect of entering into and rerouting a wide range of endeavours. If Marx's dictum that "the hand-mill gives you society with the feudal lord; the steam-mill, society with the industrial capitalist" (Marx, cited by Hui), what is it that a distributed army of Internet-connected web cameras rerouted to carry out a denial of service (DDoS) attack against websites and web hosts (Gabrys) can be said to represent? As Gabrys's afterword on these new methods of making things operational puts it: "Within the Internet of Things, what programs are to be run? Who decides which programs are to be prioritised? And how are

the conditions of the executable shifting to give rise to new problems of execution?" At a time when the iconic spectacle of execution by guillotine has been replaced with that of execution by an opaque and rapid agglomeration of black-boxed algorithms fed into remote drone operations, the task becomes that of developing "a politics appropriate to these radical modes of calculation" (Schuppli).

The term execution is often associated with death and the taking of life. Its histories include *l'exécuteur du testament*, from twelfth-century France, designating the executor of the will.[4] In such a manifestation, a specific practice of execution is already embedded in regulatory forms of bureaucracy. As Susan Schuppli highlights in her contribution on "Deadly Algorithms", the etymological and genealogical roots of the term can take on further meanings in the contemporary context of drone warfare, in which "it is only by executive decision that the US President can execute the kill order, which in turn executes a coding script that operates the remote-controlled drone, that is itself engaged in acts of summary execution". Similarly, Geoff Cox, explains how, as with the act of entering into language, there is a similar, perhaps even more overt and inherent violence to the imposition of entering into an interaction with software, particularly for the way in which "[w]ith program code, it not only symbolises but enacts violence on the thing during runtime: it quite literally executes it" (Cox). This kind of "softwar" (Angela Mitropoulos, cited in Cox) of aggression is exerted not only in overt practices of violence but also in everyday interactions with software.

It is not only that contemporary modes of execution can be seen to enact particularly strong impositions within the domains in which they operate, but also that, in many cases, these impositions come with their own forms of exception. Is it unreasonable to take an algorithm to court? What is the responsibility of an individual (human or nonhuman) in a complex computational configuration? Accountability, whether individual or collective, is buried in a mesh of technical, legal and administrative complexity. Peggy Pierrot, Roel Roscam Abbing and Femke Snelting give an example of such complexities in their chapter on the Unicode Consortium's implementation of a skin tone modifier mechanism for emoji. Their chapter highlights how the various technology corporations involved in the Unicode Consortium (such as Apple, Google and Microsoft) claim reputational victories for themselves in relation to a particular implementation, while never considering the colonial assumptions inherent within systems of encoding. As the authors highlight, in such a strategy of exception and deferral,[5] "the companies hide behind the limitations of the standard if necessary, and break out of its confines when desirable" (Pierrot, Roscam Abbing, Snelting). And if drone strikes during Obama's

presidency are one instance of executing practices, Donald Trump's election in late 2016 signals emerging ways in which politics is executed on a global scale. As (at the time of writing) Trump is ushered into office on a cresting wave of Twitter updates, election hacking controversies, algorithmically supported fake news items (so-called post-truth politics), the mainstreaming of a slew of long brewing far-right movements is taking hold in violent ways. This situation asks one to, once again, "radically rethink what it means to say 'everyone'", particularly when the de facto standpoint of the majority of the dominant corporations involved in providing the infrastructures and platforms of online expression is one of employing an "a-politicised and egalitarian discourse of diversity" (Pierrot, Roscam Abbing and Snelting).

In response to practices where various states of exception are executed, one oppositional strategy can be to uncover and create various forms of oversight and forms of accountability. Tyżlik-Carver's chapter highlights the continuous editing by many users of the Wikipedia entry on "curator", from its first registered entry at 23:19 on 6 December 2003, delivered by the IP address 131.211.225.204, to an entry in Summer 2016 that includes a fork in the main definition and describes "technology curators" as those "able to disentangle the science and logic of a particular technology and apply it to real world situations and society, whether for social change or commercial advantage". In these works we see what Tyżlik-Carver describes as the way in which executing practices of different kinds are "distributed across and performed by agents of different orders". Samson's additional contribution to the book highlights several different forms of "erasures" and the ways in which they can be seen to "execute knowledge production". Samson gives a range of examples, such as the case of University of California Davis's hiring of reputation management firms to delete an incriminating photo of a pepper spray incident on their campus, so as to avoid negative coverage of the event. Meanwhile, Hilfling Ritasdatter's essay gives a report of acts of black-boxing that sometimes unwittingly become apparent in moments of actual or potential breakdown. The anxieties and worries concerning a breakdown of global systems, caused by the Y2K bug, opens up a moment in which the many complex internal technical, economic and geopolitical relations come into focus. Hilfing Ritasdatter shows how these relations uphold the networked global economy and point towards "the neo-colonial divides" that are maintained and supported by such "anxious" executing flows.

These processes work their way across the spectrum of the political and beyond. We live and die with/in their executions. As Schuppli points out, their significance is manifest and everywhere to be seen

and experienced:

> *Algorithms have long-adjudicated over vital processes that help to ensure our wellbeing and survival, from pacemakers that maintain the natural rhythms of the heart, genetic algorithms that optimise emergency response times by cross-referencing ambulance locations with demographic data, to early warning systems that track approaching storms, detect seismic activity, and even prevent genocide by monitoring ethnic conflict with orbiting satellites.* (Schuppli)

Such executing devices are charged by existing modes of politics, just as they might enable or be reoriented to execute other potential politics. Together with them, various forms of life might be inscribed, curated, supported, destroyed or left to wither away. In Helen Pritchard's chapter on "Critter Chips", we see how organisms are held in semi-living yet enduring states by computational practices, and in Gallardo and Samson's contribution we see this in their example of how populations of brown shrimp are manipulated in ways that mutate the notion of extinction itself, highlighting neoliberalism's dependably thorough ability to financialise all aspects of life and death. Specifically, their example of shrimping involves the bringing together of the fields of computation, statistics, economics and boat design to generate a category of "commercial extinction". This is a slowly fluctuating mode of deadening as a possible mode of life — what Gallardo and Samson describe as "a comfortable form of catastrophe". This almost undead, inexhaustible drive of executable code in its ideal form is readily put into practice by neoliberal, neo-colonial and/or necropolitical (Mbembe 2003) forces in modes of operation that often veer towards exhaustion. As further evidenced in the examples of the forkbomb (Cox) and the example of the Mirai botnet DDoS attack (Gabrys), one can in these instances witness the full undead force and "ability of processes of execution to destroy the very infrastructure of the executable" (Gabrys).

In the face of any such apparent "destiny of execution" (Hilfling Ritasdatter), the direction of many of the contributions here is to suggest a politics of critique as invention, reverse engineering, intervention, repair, resistance and configuration. As the wide variety of topics and examples covered in this book acknowledge, there is an inherent excess and immanence to execution. Automation continually opens onto contingencies, breakdowns and unexpected new terrains of the executable. Similarly, execution has the quality of being both a thought experiment at the same time that it is a matter of *practising* this experiment in the world. The apt inscription and salute of executability — "Hello, world!" — captures this sense of both a putting into practice of a particular instantiation amongst

many others as well as a following of its encounters and iterations in the world. In this mesh of executing practices, the potential for configuration continues to make itself available, whether at the level of mass intervention or in the tweaking of a single line or second of code.

Notes

1. See also Fuller's brief discussion of Knuth's challenge in this same book (Fuller 2003, 17).

2. For more details, see http://atariki.krap.pl/index.php/Radiokomputer (in Polish).

3. See http://suchar.net/forum/viewtopic.php?t=15335&sid=0f308438cf03ed15f3eb13d8b6d073b7 (in Polish).

4. For a further exposition on execution, see the entry on "Execution", jointly written by several contributors to this collection, in Braidotti and Hlavajova's forthcoming Posthuman Glossary collection).

5. At the first Execution symposium (2015) held in Aarhus, Denmark, it was pointed out that such protological commands and manoeuvres on the part of contemporary modes of power can be seen to be summed up in the exultant refrain of a song from a comic opera of Gilbert & Sullivan's: "Defer, Defer, to the Lord High Executioner!" ([1885] 1992). For documentation and coverage from each of the two Execution events, see the following links: http://softwarestudies.projects.cavi.au.dk/index.php/*.exe_(ver0.1) & http://softwarestudies.projects.cavi.au.dk/index.php/*.exe_(ver0.2).

References

Animats. 2015. "What happens when you type Google.com into your browser and press enter?" *Hacker News*, January 17. https://news.ycombinator.com/item?id=8902105.

Balsamo, Anne Marie. 1996. *Technologies of the Gendered Body: Reading Cyborg Women.* Durham, North Carolina: Duke University Press

Barad, Karen. 2007. *Meeting the Universe Halfway: Quantum Physics and the Entanglement of Matter and Meaning.* London: Duke University Press.

Critical Software Thing. Forthcoming. "Execution." In *Posthuman Glossary*, edited by Rosi Braidotti and Maria Hlavajova. London: Bloomsbury Press.

Chun, Wendy Hui Kyong. 2004. "On Software, or the Persistence of Visual Knowledge." *Grey Room* 18 (Winter): 26–51.

Fritz, W. Barkley. 1996. "The Women of ENIAC." *IEEE Annals of the History of Computing* 18 (3): 13–28.

Fuller, Matthew. 2003. *Behind the Blip: Essays on the Culture of Software.* Brooklyn, New York: Autonomedia.

Gabrys, Jennifer. 2016. *Program Earth: Environmental Sensing Technology and the Making of a Computational Planet.* Minneapolis, University of Minnesota Press.

Hayles, N. Katherine. 2005. *My Mother was a Computer. Digital Subjects and Literary Texts*. Chicago: Chicago University Press.

Howse, Martin. 2013. *Diff in June*. Brescia: Link Editions. http://linkeditions.tumblr.com/howse.

Jordan, Tomasz. 1986. "Uwaga … Start!!! Radiokomputer." *Bajtek* 11 (Listopad): 28.
Kittler, Friedrich. 1997. *Literature, Media, Information Systems*, edited by John Johnston. Amsterdam: Overseas Publishers Association.
Knuth, Donald. (1989) 1991. "Theory and Practice." *Theoretical Computer Science* 90 (1): 1–15.
Mbembe, Achille. 2003. "Necropolitics." *Public Culture* 15 (1): 11–40.
Ede, Lisa and Andrea A. Lunsford. 2001. "Collaboration and Concepts of Authorship." *PMLA* 116 (March): 354–69.
Sheppard, Alyson. 2013. "Meet the 'Refrigerator Ladies' Who Programmed the ENIAC." *Mental Floss* UK, October 13.
http://mentalfloss.com/article/53160/meet-refrigerator-ladies-who-programmed-eniac.
Stengers, Isabelle. 2005. "Introductory notes on an ecology of practices." *Cultural Studies Review* 11 (1): 183.
Sullivan, Arthur, and W. S. Gilbert. (1885) 1992. *The Mikado*. New York: Dover Publications.