DATA browser 06
EXECUTING PRACTICES

Geoff Cox
Olle Essvik
Jennifer Gabrys
Francisco Gallardo
David Gauthier
Linda Hilfling Ritasdatter
Brian House
Yuk Hui
Marie Louise Juul Søndergaard
Peggy Pierrot
Andy Prior
Helen Pritchard
Roel Roscam Abbing
Audrey Samson
Kasper Ulrik Schiølin
Susan Schuppli
Femke Snelting
Eric Snodgrass
Winnie Soon
Magdalena Tyżlik-Carver

Established in 2004, the DATA browser book series explores new thinking and practice at the intersection of contemporary art, digital culture and politics.

The series takes theory or criticism not as a fixed set of tools or practices, but rather as an evolving chain of ideas that recognise the conditions of their own making. The term "browser" is useful in pointing to the framing device through which data is delivered over information networks and processed by algorithms. A conventional understanding of browsing may suggest surface readings and cursory engagement with the material. In contrast, the series celebrates the potential of browsing for dynamic rearrangement and interpretation of existing material into new configurations that are open to reinvention.

Series editors:

Geoff Cox
Joasia Krysa
Anya Lewin

Volumes in the Series:

DB 01 ECONOMISING CULTURE
DB 02 ENGINEERING CULTURE
DB 03 CURATING IMMATERIALITY
DB 04 CREATING INSECURITY
DB 05 DISRUPTING BUSINESS
DB 06 EXECUTING PRACTICES

www.data-browser.net

Executing Micro-temporality

Winnie Soon

Loading webpages, waiting for social media feeds, streaming videos and content, are mundane activities in contemporary culture. Such mundane activity includes network-connected devices that transmit and distribute data across multiple sites — referred to as data. In these scenes, data are constantly perceived as a stream (Berry 2011, 3; 2012, 388; 2013, n.p; Fuller 2003, 52), indicating characteristics of vast volume, speed of update, continuous flow and delivery. The concept of streams characterises the Internet rather than web pages (Berry 2011, 143). The web is a dynamic stream of information in which users can participate and follow. It is fast-changing and generative, data records are continuously updated and executed in a manner in which an end cannot be foreseen. There is a temporal dimension to the data stream and in today's networked communication data streams indicate events that are regarded as instantaneous in capitalised economies. The *now* that we are experiencing through perceptible streams is entangled with computational logic.

From social media feeds to playback video to mobile applications, users encounter a distinctive spinning icon during the loading, waiting and streaming of data content. This spinning icon represents an unstable streaming of the *now*. A graphical animation known as throbber tells users something is loading-in-progress, but nothing more. A similar yet very different form of a process indicator, such as a progress bar, expresses more information than a throbber. In contrast to a progress bar, which is more linear in form, a throbber does not indicate any completed or finished status and progress. It does not explain processual tasks in any specific detail when compared with a progress bar.[1] With a throbber, all that is presented is a spinning icon, perceived as repeatedly spinning under constant speed, as well as indicating invisible background activities for an indeterminate and unforeseeable timespan. If one looks up the dictionary definition of the verb "throb",[2] it is defined as a strong and regular pulse rhythm that resonates with a throbber's design and in regards to how it performs on the Internet today. But such design can be seen to oversimplify the micro-operations of networked technology, making one believe that the network is working with a certain regularity and that all data are queuing underway, thus rendering the network conditions of the *now*.

This chapter investigates data processing that takes place behind a running throbber. In particular, it examines the temporal complexity of data streams, in which data processing and code inter-actions are

operated in real-time. The notion of inter-actions references computer science's understanding of "interaction" (Beaudouin-Lafon 2008; Bentley 2003; Murtaugh 2008; Wegner 1997) as well as the notion of "intra-actions" from philosophy (Barad 2003, 2007). The term I develop here, code inter-actions, highlights the operational process of things happening within and across machines through different technical substrates, interacting with each other via running code. In contrast to the understanding of technical interaction, process emerges through "entangled agencies" (Barad 2007, 33). Barad's notion of intra-actions refers to the entanglements of material relations that are not only technically and scientifically specific, but also with mixed factors and domains of operations that are regarded as social, political, economical and cultural (2007, 232–233).

In the following session, I will illustrate how a cultural and operative reading of an abstracted form of throbber allows an examination of data streams in contemporary computational culture. This chapter will first unfold a cultural reading of a throbber, then continue with a detailed discussion and analysis of the underlying operative and technical processes. It opens up the cultural and computational logics that are constantly rendering the pervasive and networked conditions of the *now*.

A (brief) cultural reading of a throbber

With its distinct design characteristic of a spinning behaviour hinting at background processing, the throbber icon acts as an interface between computational processes and visual communication. One of the earliest uses of the throbber can be found in the menu bar of a Mosaic web browser in the early 1990s, developed by the National Center for Supercomputing Applications (NCSA), with the browser interface designed by scientist Colleen Bushell (Albers 1996; Roebuck 2011, 348–349). This throbber[3] contains a letter "S" and a globe that spins when loading a web page. This kind of a spinning throbber with the company's graphical logo can also be witnessed in subsequent software browsers, such as Netscape and Internet Explorer. While the throbber spins, it visually indicates actions are in progress. These actions, from a user's point of view, could be interpreted as the loading of web data or connecting to a website by a software browser. From a technical perspective, it involves Internet data transmission and a browser that renders the inter-actions of code. The spinning behaviour stops when a webpage is finished loading within a browser. A web browser is software able to render and display requested content, making network calls and requests, and storing data locally (Garsiel and Irish 2011). In this respect, the spinning throbber icon represents complex inter-actions of code under network conditions. A throbber,

with its spinning characteristic, can therefore be said to be rooted in, and specific to, Internet culture.

More recently, the throbber icon is no longer only attached to software browsers, appearing also on different web and mobile applications, including social media platforms in particular. The contemporary throbber transforms into a spinning wheel[4] that consists of lines or circles that are arranged in radial and circular form, moving in a clockwise direction. A throbber is animated and spun, or throbbed, with a constant rate, demonstrating a regular tempo. Each individual element of a wheel[5] sequentially fades in and out repeatedly to create a sense of animated motion. These spinning wheels appear after a user has triggered an action, such as swiping a screen with feeds in order to request the latest information. They also appear after a user has confirmed an online payment or is waiting for a transaction to complete. Perhaps most commonly of all, a throbber is seen when a user cannot watch a video clip loading smoothly over an Internet connection. As a result, an animated throbber appears as a spinning wheel on a black colour background, occupying the whole video screen while the video is buffering.

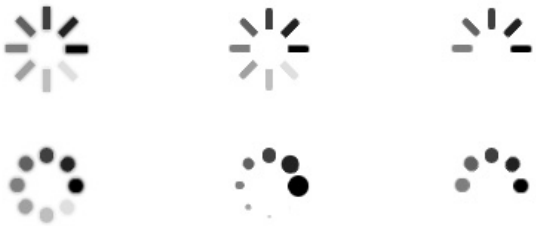


Figure 1. Throbber in the form of circles and lines, used with permission 2016.[6]

A throbber represents the speed of network traffic that is also tied to our affective states and perception of time. Emotionally, it can be frustrating to encounter buffering, as it involves interruption. Things do not flow smoothly and users become impatient in waiting for an unknown period of time or for something yet to come. Taiwanese artist Lai Chih-Sheng exhibits his throbber animated icon, titled *Instant*[7] (2013), with a minimalistic presentation, expressing the relation between waiting and time. This waiting is considered as unproductive, in that it consumes time. As artist-researcher James Charlton describes it: "It is a gaze that goes beyond the screen to an event not yet here" (2014, 171). The loading time of the throbber appears wasted and unproductive, as it is often associated with the perception of slowness of a network.

On September 10, 2014, a campaign called "Internet Slowdown day"[8] was launched as part of the "Battle for the Net", promoting net neutrality and Internet freedom. Customised loading icons, similar to a throbber, were put up on different websites, symbolising the potential impact of controlled traffic that would be implemented by Internet Service Providers in the name of increasing profit. In other words, the campaign argued for Internet speed equality across all websites and that no unequal conditions, such as fast-lane traffic, should be given to any prioritised website. More than 10,000 corporations showed support by putting up self-designed throbber icons. As is evident in this context, the throbber has a significant and symbolic meaning within cultural and political realms.

In contemporary art,[9] the throbber as cultural icon is remade by artist Aristarkh Chernyshev, showing the spinning behaviour through customised LEDs in a physical installation. The LEDs formulate the word "loading", circulating in a motion directly reminiscent of a spinning throbber. Chernyshev's artwork *LOADING* (2007)[10] aims to present this icon and its data exchange process as cultural phenomena, with the cultural icon of a throbber expressing various dimensions of time — from the loading time of a browser to the regular tempo of a spinning throbber to the slowness of the Internet network — in understanding data streams. Beyond different cultural instances, however, the operative and technical dimensions of a running throbber should not be undermined, as they can provide a specific perspective for further understanding how the now is being organised computationally as streams.

Drawing from a method proposed by Wolfgang Ernst in the field of Media Archaeology, the meaning of data streams can be analysed and understood via an application of a "cold gaze" upon data streams. Ernst's approach is used to engage with the mechanical and operational logic of computation, and the method of "cold gaze" aims to describe cold facts in a distinctly material-oriented, as opposed to narrative-based, approach (Parikka 2011, 2012; Ernst 2013). In taking into consideration the operative and technical perspectives of network transmission, Florian Sprenger provocatively argues that the concept of stream is a metaphor. He says:

> *The network structure of today's communication channels and of their information stream is often understood as providing a direct connection between users and services or between two communication partners, even though there cannot be any direct connections on digital networks. The metaphor of the flow conceals the fact that, technically, what is taking place is quite the opposite. There is no stream in digital networks.* (Sprenger 2015, 88–89)

Sprenger highlights the possible misconception of a flow or a stream, suggesting that there is a gap between the experienced and operative streams. He reminds us that two widely used concepts — flow and stream — in digital media are metaphors that potentially mislead anyone looking to understand the actual technical processes that take place beneath a stream. Drawing upon Ernst's (2013, 186–189) notion of micro-temporality, the focus of such approaches is with the nature and operation of signals and communications, mathematics and digital computation within its deep internal and operational structures. The added prefix "micro", therefore, addresses the micro-operative processes that are not apparent within an immediate human register. Ernst's notion of micro-temporality draws (after Foucault) on the concept of discontinuity (2006, 105). In Foucault (1972, 3), discontinuity offers an alternative perspective to understanding knowledge beyond its stable form of narration and representation. Both Foucault and Ernst use discontinuity as a means to examine the gaps and ruptures of things that go beyond signs or representational discourses.

To bring together concepts of discontinuity and micro-temporality is to offer an alternative perspective in examining streams behind a planetary scale global economy which renders the *now*. Streams can be understood as highly capitalised and as operating in massive scales under globalised processes that disseminate into every part of the world as cultural and economic phenomena. In the words of Peter Osborne, the *now* "is primarily a global or a planetary fiction" (2013, 26). Thus, the notion of discontinuous micro-temporality highlights the micro-processes and gaps of a stream that is manifested within networked presence-oriented feeds and their regular interruptions by a throbber. The concept of discontinuous micro-temporality points towards the temporal dimension of streams that present the *now*. The following section will take a micro-temporal analysis to foreground the notion of discontinuous micro-temporality that takes into account operative processes.

Micro-temporal analysis

Following the Von Neumann Architecture that was first initiated in 1945, mathematician and physicist John von Neumann designed a computer architecture consisting of a processing unit that contains an arithmetic logic unit, a control unit and a memory unit for performing arithmetic operations, operational sequence control and data and instruction storage respectively, also known as a stored-program computer (von Neumann 1945, 1–2). In this setup, a central clock[11] coordinates these units, executing computer instructions in a precise manner.

The appearance and disappearance of a graphical throbber is rendered by code, instructing when a throbber should be displayed on a screen. However, computer instruction is more than source code. In Computer Science and Engineering, the "Fetch-Execute cycle" is used to describe how a Central Processing Unit (CPU) performs code instructions through a series of steps that are executed within clock cycles (Burrell 2004, 135; Frabetti 2015, 153). The high-level instruction breaks into many micro-instructions by fetching and executing values from and in the memory space. The micro-instructions are highly ordered. The instruction pointer (also known as program counter) is used to keep track of the instruction sequence. This pointer is incremented after fetching an instruction and storing the memory address of the next instruction to be executed. The computer will continue repeating the cycles that fetch instructions and data from memory and then execute them one after another in sequence until the final instruction is reached (see also Frabetti 2015, 150–159). In short, executing code instructions involves the reading and writing of memory,[12] generating a sequence of micro-operational steps and the actual computation. The appearance or disappearance of a throbber on a screen is not an exception. All of the code instructions are operated across on/off states, generally known as "flip-flops" and logic gates used to store and control data flow. Underneath a graphical throbber is the inter-action of data, code and micro-instructions. The micro-temporality of instructions is driven by the internal clock as there are things that have to be done exactly at a specific time. Importantly, the machine clock forms a basic infrastructural activity of contemporary technology, organising and maintaining the sequences and components of computation that are essential in performing operational tasks. This micro-perspective allows us to be attentive to how time is structured and organised computationally and differently.

Packet switching and data buffering

Networked data are streamed over a technological network. This also relates to how data transfers and operates geopolitically across devices that are constrained by structures, infrastructures and "micro-decisions" (Sprenger 2015) along a transmission process. The following discussion will focus on the processes of packet switching and data buffering that are operated behind a running throbber.

In the late 1960s, the world's first packet switching network, called the ARPANET, was introduced, laying the groundwork that led to the development of the Internet as it has developed today. The concept of packet switching was fundamental to understanding how data are organised and flow. A data stream was chopped into smaller blocks as "packets", which were then sent via a communications channel in

and through different routes, rates and sequences, known as packet switching (Baran 2002). Between the two connection points — sender and receiver — data, indeed, does not have a direct connection. According to Paul Baran, one of the inventors of the packet switched computer network, real-time connections between sender (transmitting end) and user (receiving end) are an illusion. Instead, the fast-enough data rate gives only a *sense* of real-time connection between a sender and receiver. Fundamentally, the routing of a data packet transmits through different sites. Although a selected path is based on "adaptive learning of past traffic" (44), there are real-time decisions that have to be made to locate the shortest path[13] due to the dynamics of network conditions. In other words, data travels "via highly circuitous paths that could not be determined in advance" (43).

It is worth noting that data packets pass through intermediate devices like gateways, switches and routers in their journey. According to the Protocol specifications (RFC 793 and RFC 791), there is a field called "Time to Live" (TTL) that limits the lifespan of data within a connection (Postel 1981b, 51, 1981a, 14). Data packet routing means that a connection between sender and receiver contains multiple switching computers and a route is made up of multiple "hops".[14] TTL is defined as the number of hops that a packet has to pass through before reaching its destination. This also means that if a packet passes through more than a defined number of hops, that particular packet is being discarded, alluding to the time to die, as opposed to live. Therefore, each packet has its own lifespan. The idea behind having the TTL field is to prevent any instances of endless circulating of data packets within the network. These decisions are monitored and executed in real-time. This real-time execution is similar to what Wendy Hui Kyong Chun describes within the context of hardware and software systems in which computation responds to the live condition. She says,

> *[H]ard and software real-time systems are subject to a 'real-time' constraint — that is, they need to respond, in a forced duration, to actions predefined as events. The measure of real time, in computer systems, is its reaction to the live — its liveness.* (Chun 2008, 316)

The notion of liveness can be understood as the decisions and reactions that are required to execute beneath various real-time constraints. To Chun, liveness is expressed at the temporal level in which a system is required to react and respond according to its user input and output. But in the case of technological networks, the response may not include direct human intervention, and machines take charge of decisions and in real-time and responses in a forced duration. The micro-temporality of a stream involves "micro-decisions" (Sprenger 2015) as well as

interruptions in real-time. Every micro-decision, the routing decision via multiple hops for example, takes time. Decisions are made not only in real-time but also in a micro-temporal interval. This also applies to the process of data buffering; we normally understand this by seeing a throbber that interrupts a stream. What then are the micro-decisions involved in data buffering?

A buffer is understood as a temporal storage that usually stores a small amount of data in physical memory. While some data are stored in a buffer, other segments of data are being read and processed. This also means that software applications are not required to wait for the entire media file to be downloaded. "Just in Time" (JIT) delivery is used in streaming media, allowing for the playback of partially received data temporarily stored in the client's buffer (Pereira and Ebrahimi 2002, 260). In this sense, both the playback of buffer data and the receiving of the remaining data can be made simultaneously (and, in addition to the case of video and audio, this is also commonly experienced in loading any relatively large size file, such as a PDF or an image within a browser). The buffer is where software applications, such as a browser or media player, access the input data and process it as output data. In other words, the processing of data consists not only of the transferring part, but rather, as Ernst reminds us, through "a coupling of storage and transfer in realtime". He continues, "[w]hile we see one part of the video on screen, the next part is already loaded in the background" (Ernst 2006, 108). More precisely, the viewer is not watching the content as data arrives, instead, the viewer is watching the processed data that has arrived and stored in the buffer. This process of temporal storage and playback gives us an understanding of the relation between buffer and streams, in which there is latency between data arrival (from the network), data storage (within internal memory) and data processing (inside a machine) at micro-time intervals. Streaming is essentially "achieved by buffering the transmitted data before the actual display" (Meinel and Sack 2013, 780). A throbber is entangled with this latency, inter-acting with different pieces of data in different ways.

Ideally, the "buffer empties itself at one end just as quickly as it fills up at the other end", as described by Christoph Meinel and Harald Sack (783). If there is transmission delay that is within a threshold time *t*, it is regarded as unnoticeable in playback. However, if the delay of the individual segment exceeds the threshold time *t*, a throbber will then display. A program performs to read and process the buffer but the data has not arrived yet, and this gap and rupture will lead to the appearance of a throbber. This is the instance in which we can perceive and experience the discontinuous micro-temporality.

Normally, a throbber is seen when loading a big chunk of data, which is commonly seen in video sites, mostly due to the instability or low bandwidth of a network that causes the delay of data segment arrival (exceeds the threshold time *t*). Buffering is highly related to time as it allows different rates to occur simultaneously, decoupling "time dependencies" between the input and output of data (55). As a result, data can be consumed and processed at a different rate by program applications. Data, in the case of streaming, is actively and constantly being stored (written) and removed (read) in the buffer at different speeds and rhythms, oscillating between the invisible and visible. The micro-temporality of buffering transforms the space of a buffer that works with both internal and external data. This buffer space, as a site of inter-actions, contingently and temporally performs variations. Although what has been written in the buffer will be automatically read and processed, technology does not guarantee that all the data are written in the buffer.

The absence of data

Dropped frames (frames of video that are dropped during playout) are a relatively common experience in real-time communications and video streaming. Dropped frames impact upon the user's viewing experience because of frames that disappear within a perceivable continuous stream. When an audio-visual is played back at the receiver's side, this introduces gaps in the stream and it is able to produce glitches or jittery audible effects. This is different from displaying a throbber on a screen, where nothing can be seen on a screen despite the animated graphic. When experiencing dropped frames, one can still see or hear something, but just not necessarily in good quality.

In some situations, the issue of dropped frames is seamless because it does not create significant quality degradation. Such visible and invisible dropped frames are caused by packet loss, the absence of certain parts of data during data transmission across nodes throughout the journey. Indeed, packet loss is highly relevant to the notion of micro-temporality. According to James F. Kurose and Keith W. Ross, the delay time for transmitting data does not only include "store-and-forward" in each buffer nodes, but also "queuing delays" that are subjected to network congestion and are not predictable in advance (2013, 25). Packets are required to queue up and wait for the transfer while the network is congested. Under streaming conditions, data are continuously transmitted across multiple sites. However, the amount of buffer space is limited at each site, which means a newly arriving packet potentially has no space to be stored in while the stored packet is still queuing for its next routing. In this

situation, "packet loss will occur — either the arriving packet or one of the already — queued packets will be dropped" (25).

The robust design of network protocols consists of an automatic mechanism to detect and trigger retransmission for packet loss. However, for real-time conversational applications and media streaming platforms for live concerts, such as Skype and YouTube, delay time for each packet is a critical issue as the transmission demands to be continuous. Both conversations and live concerts are unceasing. On the one hand, the absence of data is crucial as packet loss is related to the degradation of quality, and it could immediately impact the visual or audio quality in a live environment. On the other hand, if data arrives with significant delay, the application design at the receiver's end is then required to determine if such data will still make sense in playback, in particular where conversation and data are constantly played-back as a stream. In deciding whether the data should be played-back or ignored, acceptable latency becomes a decision that is inscribed in the software and platform design. During streaming conditions, a throbber will be seen for a weak connection (as for the case of Skype conversation). A serious data loss may even result in the automatic termination of a connection — which also means the tolerance is unacceptable from the point of view of software design. The technical consequences of data loss is nothing new if one has used Skype or other communication applications like what's app, weChat or Line, in which it is not uncommon to have the experience of glitches or jitter effects, as well as a throbber display on a screen. But what is of concern here is rather the cultural implications of these absent data, or the potentiality of packet loss at any moment of time.

Here the absent data requires our attention. Firstly, the absence of data might be caused by a voluntary condition. It is possible for an application to discard late-arriving data that are within acceptable latency because it is insignificant to the entire user experience. Secondly, due to the buffer capacity, data loss can occur anytime and at any sites during the entire journey of a data transmission. Last, but not least, when the network bandwidth cannot match the application's processing rate, there will be data loss.[15] As a result, not all data are treated equally and able to arrive at the destination and take a perceptible form. Even though the presence of a stream is mediatised as audio and visuals through a screen, there is still the possibility of absent data. The absence, although it cannot be mediatised in its perceptible form at the receiver's end, is implied in the presence of streams, in which conversation or video playback is kept running. The point is that the mundane activity that we wait and stream through a screen is loaded with unperceivable gaps.

To explain further, the logic of buffering and data processing are constantly performed through the presence and absence of data. A display of a throbber presents another reality, a reality that is conflated with an invisible material infrastructure and the absence of material substrates. Furthermore, a throbber and its underlying data buffering involve discrete-time signaling—the milliseconds of time lost and the absence of data-presenting multiple realities which lie at the heart of time-dependent logics. Therefore, reality is not only a matter of continuous flow and the immediacy of a stream. Taking account of materiality, such a notion of reality refers neither to the symbolic meaning of content, the feeling of presence or the immediacy of data delivery, but rather a tension is expressed between continuity and discontinuity through the performativity of code. That is to say, when taking into account packet loss, the liveness or *nowness* of a stream is about an absent present. The notion of discontinuous micro-temporality explicates the invisibility of computational culture by shifting our attention from the cultural understanding of a throbber and what is visible on a screen to invisible micro-events that are running in the background, events that are not separated but entangled as absent present.

Absent data are rarely mentioned in the commercial products that frame contemporary digital culture, inasmuch as it possibly relates to quality degradation or may be regarded as not noticeable. Within a stream, there are these discontinuous forces that constitute the continuous presence. Sometimes the forces appear to be strong, yet at other times they are weak; in some cases more visible, and at other times unnoticeable. The notion of discontinuity pays attention to the gaps, ruptures and pauses that are interwoven within the continuous flow of a data stream. From the display of a running throbber to its disappearance while a stream is presented, *discontinuous micro-temporality* highlights the forces and presence of micro-decisions and micro-interruptions that reconfigure the nowness or liveness of a stream.

Conclusion

A stream is manifested into continuously updating feeds, passing through hops and sites, which in part defines the now. The mundane throbber calls for a critical attention towards mediated processes not only at a planetary scale, but also at the micro-temporal level of operations, including clock cycles, instructions execution, packet switching and data buffering, which exhibit micro-decisions and micro-interruptions. The notion of discontinuous micro-temporality takes into account the micro-processes, gaps and ruptures and, more importantly, the absence of data that renders present realities.

This sheds light on the understanding of streams in computational culture, in particular, on how time is processed and organised to present the *now* under live conditions.

The existence of a throbber is a by-product of a commercial application that informs users to wait for an unknown period of time. Through the use of a throbber in developing various services — such as live streaming, social media platforms, data and transactional applications — this cultural icon offers a critical space for understanding how the *now* is being made operative. A throbber is a cultural phenomenon that appears in almost every application that operates within a live computational environment. A throbber is not only a technical or functional object but also entangled with other cultural and micro-processes. This chapter explicates the computational logic behind a throbber as well as the real-time dynamics of computational networks and, hence, the rendering of the pervasive and networked conditionings of *nowness*.

Acknowledgements

The concept of this paper was developed through the seminars and conferences on 'execution' in 2015 and 2016. I am thankful to the team of Critical Software Thing and the team of respondents, including Geoff Cox, Christian Ulrik Andersen and Femke Snelting, as well as the participants of the execution conference for their comments on the earlier concept of this paper.
I am especially grateful to the editors of this book, providing insightful and constructive feedback. Finally, an earlier version of this chapter has been published in ISEA2016 Conference Proceedings, and I thank the reviewers and Jane Prophet in providing their comments.

Notes

1. Computer operations are usually explained in conjunction with the use of a progress bar, for example, the transferring and copying of specific files and directories, or illustrating installation procedures.

2. See: http://www.oxforddictionaries.com/definition/english/throb.

3. The mosaic throbber also allows user to click on it to stop loading a webpage (Roebuck 2011, 348).

4. The use of lines that indicates the progress activity of a computer can be found in the early operating system of Unix that consists of few string characters as '[', '—', '\', '|', '/', ']' (Roebuck 2011, 349).

5. Coincidently, the visual design of a throbber is similar to the design of early wristwatches (with crystal guards) that were made for soldiers in World War I. Both include the concept of a wheel in the form of circles or lines of petal shape. See: http://www.oobject.com/category/earliest-wrist-watches/.

6. Source: "18 CSS3 and jQuery Loading Animations Solution." Design Modo, May 30, 2015, http://designmodo.com/css3-jquery-loading-animations.

7. See: https://www.facebook.com/ESLITE.PROJECTONE/photos/?tab=album&album_id=437623016346200.

8. For more details, see: https://www.battleforthenet.com/sept10th/.

9. Other artists have also explored this throbber icon. For example, artist Gordan Savičić explores the perception of time through his work *Loading* (2009), that turns an ordinary windowpane into a screen (Savičić 2009). Alongside this chapter, I have also developed a project called *The Spinning Wheel of Life* (2016)

that explores the micro-temporality of computation (Soon 2016).

10. See: https://festivalenter.wordpress.com/2009/04/09/electroboutique-by-alexei-shulgin-roman-minaev-aristarkh-chernyshev/.

11. Thanks to Brian House who first introduced the concept of computer clock to me in the *.exe (ver0.1) workshop (House 2015).

12. Memory is used here in a broad sense that includes computer main memory, instruction register and memory buffer register, etc.

13. For more details about the determination of the shortest path, see Meinel and Sack (2013, 350–352).

14. A hop refers to "the leg of a route from one end system to the nearest switching computer, or between two adjacent switching computers, or from the switching computer to a connected end system" (Meinel and Sack 2013, 451).

15. For example, a 50% data loss is encountered when a network has only a maximum bandwidth of 5 Mbps and the application requires 10 Mbps.

References

Albers, Michael C. 1996. "Auditory cues for browsing, surfing, and navigating." In Frysinger, S.P. and G. Kramer (eds), *Proceedings of the 3rd International Conference on Auditory Display (ICAD 1996).* Palo Alto, California: ICAD.

Barad, Karen. 2003. "Posthumanist Performativity: Toward an Understanding of How Matter Comes to Matter." *Journal of Women in Culture and Society*, 28(3).

——. 2007. *Meeting the Universe Halfway: Quantum Physics and the Entanglement of Matter and Meaning.* Reprint. Durham: Duke University Press.

Baran, Paul. 2002. "The Beginnings of Packet Switching: Some Underlying Concepts." *IEEE Communications Magazine* 40 (7): 42–48.

Beaudouin-Lafon, Michel. 2008. "Interaction Is the Future of Computing." In *HCI Remixed: Reflections on Works That Have Influenced the HCI Community*, edited by Thomas Erickson and David W. McDonald, 263–266. Cambridge, MA, London: The MIT Press.

Bentley, Peter. 2011. "The Meaning of Code." In *Code: The Language of our Time*, edited by Gerfried Stocker and Christine Schöpf, 33–36. Linz: Hatje Cantz, 2003.

Berry, David M. *The Philosophy of Software Code and Mediation in the Digital Age*. Basingstoke: Palgrave Macmillan.

——. 2012. "The Social Epistemologies of Software." *Social Epistemology* 26 (3–4): 379–398.

——. 2013. "Introduction: What is code and software?" In *Life in Code and Software: Mediated Life in a Complex Computational Ecology*. Open Humanities Press.

Burrell, Mark. 2004. *Fundamentals of Cmputer Architecture*. New York: Palgrave Macmillan.

Charlton, James. 2014. "Post Screen Not Displayed." In *Post-Screen: Device, Medium and Concept,* edited by Helena Ferreira and Ana Vicente, 170–182. Lisbon: CIEBA-FBAUL.

Chun, Wendy Hui Kyong. 2008. "On 'Sourcery,' or Code as Fetish." *Configurations* 16 (3): 299-324.

Ernst, Wolfgang. 2006. "Dis/continuities: Does the Archive Become Metaphorical in Multi-Media Space?" In *New Media, Old Media: A History and Theory Reader*, edited by Wendy Hui Kyong Chun and Thomas Keenan. New York, London: Routledge.

——. 2013. *Media Archaeology: Method and Machine Versus History and Narrative of Media*. Edited by Jussi Parikka, Digital Memory and the Archive. Minneapolis: University of Minnesota Press.

Foucault, Michel. 1972. *The Archaeology of Knowledge and the Discourse on Language*. New York: Pantheon Books.

Frabetti, Federica. 2015. *Software Theory: A Cultural and Philosophical Study (media philosophy)*. London: Rowman & Littlefield International.

Fuller, Matthew. 2013. *Behind the Blip: Essays on the Culture of Software*. New York, London: Autonomedia.

Garsiel, Tali, and Paul Irish. 2011.

"How Browsers Work: Behind the Scenes of Modern Web Browsers." Last modified August 5. http://www.html5rocks.com/en/tutorials/internals/howbrowserswork.

House, Brian. 2015. "Sound and The Heat of The Cut." http://softwarestudies.projects.cavi.au.dk/index.php/Exe0.1_Brian_House.

Kurose, James F., and Keith W. Ross. 2013. *Computer Networking: A Top-Down Approach*. Pearson Education.

Laplante, Philip A. 2000. *Dictionary of Computer Science, Engineering and Technology.* CRC Press.

Meinel, Christoph, and Harald Sack. 2013. *Internetworking: Technological Foundations and Applications*. Berlin: Springer.

Murtaugh, Michael. 2008. "Interaction." In *Software Studies: A Lexicon*, edited by Matthew Fuller, 143-148. Cambridge, MA: The MIT Press.

Osborne, Peter. 2013. *Anywhere Or Not At All: Philosophy of Contemporary Art*. London: Verso.

Parikka, Jussi. 2011. "Operative Media Archaeology: Wolfgang Ernst's Materialist Media Diagrammatics." *Theory, Culture & Society* 28 (5):52–74.

——. 2012. *What is Media Archaeology*. Cambridge: Polity Press.

Pereira, Fernando, and Touradj Ebrahimi. 2002. *The MPEG-4 book*. Prentice Hall.

Postel, Jonathan. 1981a. Internet Protocol - Darpa Internet Program Protocol Specification (RFC 793). Information Sciences Institute.

——. 1981b. Transmission Control Protocol—Darpa Internet Program Protocol Specification (RFC 791). Information Sciences Institute.

Roebuck, Kevin. 2011. *Virtual Desktops: High-Impact Strategies—What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. Dayboro: Emereo Publishing.

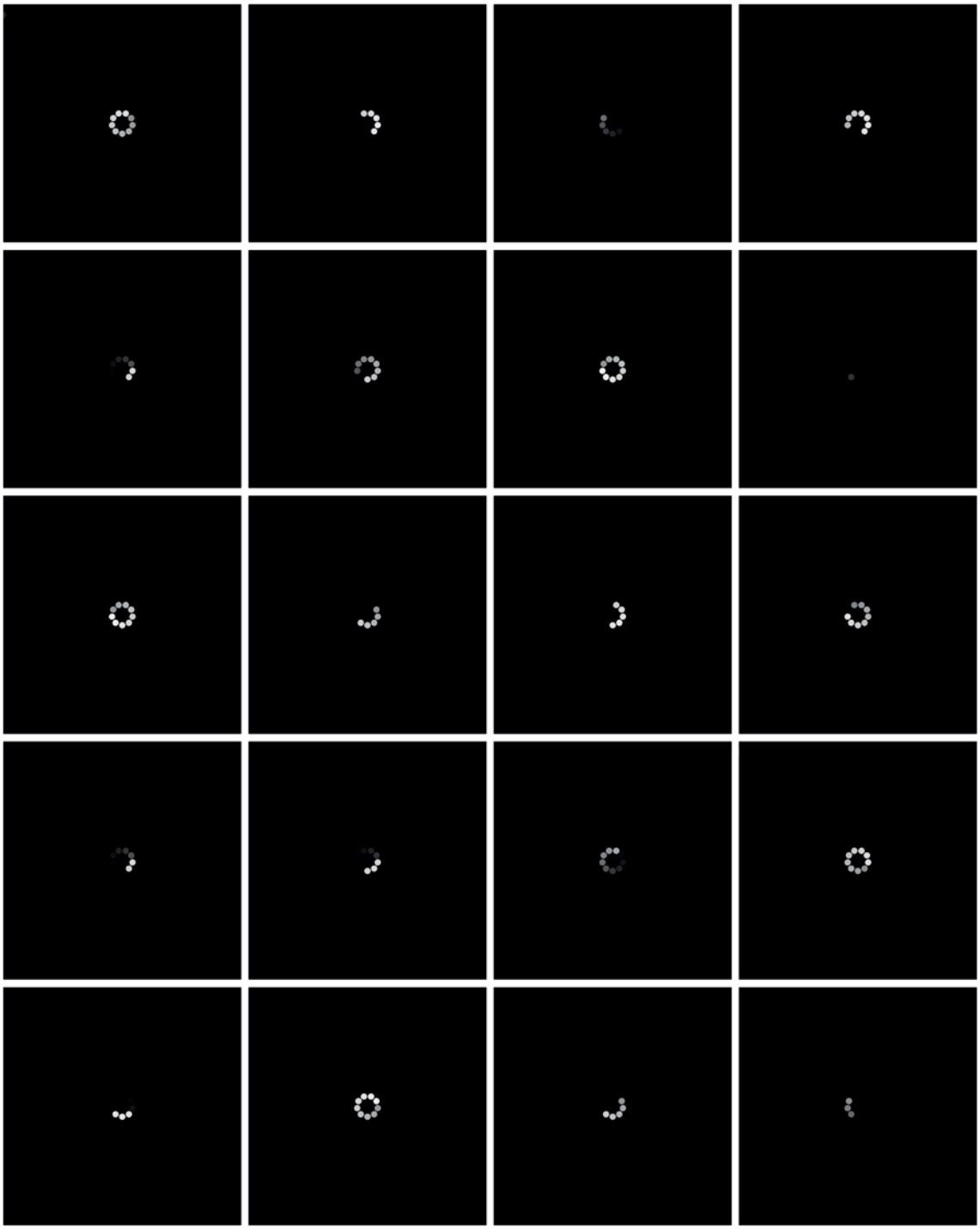
Savičić, Gordan. *Gordan Savicic—fleshgordo vs. frescogamba*, http://www.yugo.at/processing/?what=loading.

Soon, Winnie. 2016. "The Spinning Wheel of Life (work-in-progress)." *Digital Art and Technology*. http://siusoon.net/home/?p=1407.

Sprenger, Florian. 2015. *The Politics of Micro-Decisions: Edward Snowden, Net Neutrality, and the Architectures of the Internet*. Lüneburg: Meson Press.

Von Neumann, John. 1945. "First Draft of a Report on the EDVAC."

Wegner, Peter. 1997. "Why Interaction is More Powerful than Algorithms." *Commmunications of the ACM* 40 (5): 80–91.



Winnie Soon, *The Spinning Wheel of Life* (2016)