eludamos
Journal for Computer Game Culture
Vol. 8, No. 1 (2014)

http://www.eludamos.org

"Tap, tap, flap, flap." Ludic Seriality, Digitality, and the Finger

Till A. Heilmann

Eludamos. Journal for Computer Game Culture. 2014; 8 (1), pp. 33-46

“Tap, tap, flap, flap.” Ludic Seriality, Digitality, and the Finger

TILL A. HEILMANN

Flappy Bird

In early 2014, a small game for Apple iOS and Google Android devices called *Flappy Bird* enjoyed a short-lived but spectacular success. The game by Thai indie developer Dong Nguyen had been published almost a year before but for months had remained one of thousands of little-noticed titles on Apple’s App Store. Then, almost overnight in December 2013, it shot to the top of the download charts and started earning Nguyen $50,000 a day from in-game advertisements. Suddenly, *Flappy Bird* was all over the news sites and social media. Players found the game frustratingly difficult but highly addictive. Accusations arose that Nguyen had plagiarized from other games and that he had used bots to promote his title. Amidst a rising media frenzy, the reclusive developer eventually pulled the game from Apple’s and Google’s online stores in March 2014. In the guise of countless clones, however, *Flappy Bird* remains popular to this day (so much so, in fact, that many of these clones serve to spread malware; cf. McAfee 2014).

The story of *Flappy Bird* has been told many times. The purpose of this paper is not to give another account of the game’s “rise and fall” (see Kushner 2014, Rigney 2014), to try to explain its success (see Juul 2014), to analyze its carefully balanced mechanics (see Stuart 2014), to comment on its roots in game history and its aesthetic allusions (see Schreier 2014), or to give an existentialist critique of the gameplay (see Bogost 2014). Instead, *Flappy Bird* is approached here as a paradigmatic example of seriality in videogames in order to discuss some theoretical aspects of digitality, both in games and at large. Formulated in the most general way, the question this paper addresses is this: What is “digital” about digital media? Let us begin our investigation of this question with a detailed description of *Flappy Bird*.

After selecting “START” on the splash screen (the only other options being “SCORE” and “RATE”), you are presented with what might be called the instruction screen: Viewed from the side, you see a small yellow bird—“Flappy Bird” himself—against a backdrop of turquoise sky. Flying from left to right, Flappy Bird is a roughly ball-shaped creature with huge white eyes (of which you can see only one), a big red beak that looks more like a pair of painted lips, and small wings (of which, again, you can see only one). The ground at the bottom of the screen is a straight band of striped green with a thin black line on top. The horizon is lined with treetops, skyscrapers, and a sea of white clouds. At the top of the screen, there is a blocky “0” in black and white (apparently some sort of counter) and a pause button in the upper left-hand corner. While the game waits for your input to begin, Flappy Bird flaps its small wing(s), lightly bouncing through the air while the ground scrolls by endlessly. In the center of the screen sits a grey, motionless “ghost” double of the bird. Below it there is a black upward-pointing arrow and a white cartoon hand with the index finger also pointing upwards, next to which is a red guide-post arm that reads “TAP.” Right above Flappy Bird and its ghost, words spelled in big letters tell you to “Get Ready.”

Now you touch the screen with your finger (or your thumb[1]) and Flappy Bird’s “ghost” double, along with the instructional signs and messages, disappear as the game begins: After an initial bounce upwards and without further action from your side, Flappy Bird stops flapping its wing(s) and starts to dive steeply towards the ground. As you were told to do by the game’s instructions, you quickly tap the screen and the bird reacts by moving its wing(s) once, sending it up a bit. Every time you tap the screen, the bird’s wing(s) flap and it gains some altitude. When you don’t, it loses height fast. Just as you have grasped this basic mechanism and have learned to keep Flappy Bird from crashing, green pipes appear from the right side of the screen. Protruding up from the ground and hanging down from the top of the screen, each pair of pipes leaves a small gap in between. Coming one after the other, the evenly spaced pairs of pipes are the only enemy Flappy Bird faces—a simple yet dangerous obstacle. When Flappy Bird hits a pipe (and the game’s collision detection is unforgiving), it falls to the ground with a whacking sound and the game ends. Another tap lets you play again, and *Flappy Bird* starts over from the beginning.

Whenever Flappy Bird successfully passes a pair of pipes, the counter at the top of screen is increased by one. Manoeuvering the bird through the gaps sounds easy enough, but due to the physics of bird flight involved and the carefully chosen width of the pipes and the spaces between them, it proves extremely difficult. You will probably need half a dozen tries or more just to master the first pair of pipes. And it doesn’t get much better after that. Still, no matter how many pipes you pass, nothing ever changes except the random vertical position of the gaps and the steadily increasing counter. All you see is more and more pipes, and all you do is tap and tap again and again. This most simple of gestures—touching the screen with the tip of a finger, it doesn’t even matter where you put it—is the only interaction you have with the game. Reduced to a single rudimentary mechanism and having no other goal than for Flappy Bird to stay airborne and alive as long as possible, the game forces you to concentrate on the temporal pattern of minute bodily movement, the right rhythm of the hand, the constant drum of your finger. It consists entirely of a (potentially infinite) series of minimal inputs to and outputs of the machine: Tap, tap, flap, flap.[2]

Ludic Seriality

A serially organized input and output circuit between man and machine is a basic trait of all videogames. On the most fundamental level, “seriality” simply means that any kind of minimally complex behavior necessarily unfolds as a chain of actions in time. In videogames, these chains of actions constitute specific temporal “patterns of repetition and variation” (Denson and Jahn-Sudmann 2013, p. 13) which are characteristic of different sorts of gameplay and genres (e.g. shoot ‘em up, text adventure, or city-building game). Denson and Jahn-Sudmann have proposed the term “ludic seriality” to designate the different aspects of seriality pertaining to (video-)games and to call attention to the various “aesthetic forms and the cultural practices of serialization as they are articulated in and around interactive digital media” (pp. 10-11). Denson and Jahn-Sudmann distinguish three categories of ludic serialization: para-ludic, inter-ludic, and intra-ludic seriality. Whereas para- and inter-ludic seriality occur between individual games (e.g. in the form of game sequels and series) and in relation to their larger cultural framework (e.g. as games adapted from other media such as comic or film), intra-ludic seriality manifests in recurrent ludic elements and structures within games: power-ups, lives, boss fights, levels, and so on.

Let us try to apply Denson and Jahn-Sudmann's categories to an analysis of *Flappy Bird*'s serial character. While no obvious cases of para-ludic seriality come to mind, examples of inter-ludic seriality abound. The origins of the game's concept can be traced back to early arcade titles like *Defender* (Williams Electronics 1980). More specifically, though, *Flappy Bird*'s simplistic execution of the endless side-scroller formula has an unmistakable forerunner in a little-known Flash-based browser game called *Helicopter Game* (SeeThru.co.uk 2000), a tie-in to a BBC TV show. Particularly striking are the similarities to *Piou-Piou contre les cactus* (Zanorg 2011), a side-scroller in which you have to navigate a small yellow bird with big red lips (!) through a row of green cacti growing up from below and hanging down from above (!). Indeed, there have been repeated accusations of plagiarism, all of which Dong Nguyen has firmly denied.[3] In addition to the overall design of the game, certain of its elements also seem to be heavily inspired by earlier games. The green pipes Flappy Bird has to evade look a lot like the pipes from Nintendo's *Super Mario* series. And the sound *Flappy Bird* makes when you pass a pair of pipes is very reminiscent of the famous *Super Mario* coin sound effect. All in all, *Flappy Bird* takes up and continues a long series of genre titles on the one hand and a distinct tradition of 8-bit aesthetics rooted in early Nintendo culture on the other hand. And at least as numerous as the serial forms it carries on in this way are the inter-ludic serializations that refer back to *Flappy Bird*—particularly, the dozens of clones that appeared for iOS and Android devices and for web browsers after Ngyuen removed *Flappy Bird* from the iTunes App Store (cf. Tassi 2014).

Of course, *Flappy Bird*'s serial character shows strongest on the intra-ludic level. Seriality, one could say, is the kernel of the game. It is, in fact, the one and only principle of gameplay. To play *Flappy Bird* means to perform a series of taps on the screen in order to guide the bird successfully through a series of pipes so as to increase the counter one by one over the series of natural numbers. Every pair of pipes, every tap of the finger, every beat of the wing and every stepping of the counter is just like the one before, the only variation being in the position of the gaps between pipes, the cadence of the finger, the bird's altitude and pitch and the value of the counter. In concept, aesthetic, and practice, *Flappy Bird* presents us with seriality stripped down to a minimal yet enjoyable form.

Operator Action

The simplicity of *Flappy Bird*'s gameplay certainly contributes to its addictive appeal. It also makes evident an important point about play in videogames. "Play itself, we must recall, is an essentially serial activity" (Denson and Jahn-Sudmann 2013, p. 8). Seriality is not an option you can choose (or choose to ignore) when designing or playing a game. It is the very structure of play. "*[V]ideo games are actions,*" as Alexander Galloway (2006, p. 2) puts it. "To understand video games, then, one needs to understand how action exists in gameplay" (p. 3). What appears as a "*unified single phenomenon*" (p. 5) in the course of a game can be analyzed into actions caused by the player—or *operator actions*, as Galloway calls them, e.g. moving an avatar through the game world—and actions caused by the hardware and software used to play—called *machine actions* by Galloway, e.g. movements by enemies. A particular play of a game, then, can be described as the series of operator and machine actions that constitute the gaming process from beginning to end.[4]

The word "series" in the last sentence implies more than just a succession of actions. It also means repetition. For the choices a player has in playing a videogame are never without limit. The range of operator actions can be very narrow, as is the case with *Flappy Bird*, or it can be very broad, as is the case with *Flight Simulator X* (Microsoft Game Studios 2006)—or it can be anywhere in between. But just as the number of game rules or mechanisms is always finite, so is the number of possible operator actions. A title like *Flight Simulator X* may tend to obscure this fact while one like *Flappy Bird* rather exposes it. In either case, though, the player is only free to choose from a finite set of options. In *Flappy Bird*, there is only "tap to flap" (and "don't tap to dive," if you want to count inactivity as an option). In *Flight Simulator X*, you can pilot a Boeing 747 aircraft around the world using dozens of controls in the simulated cockpit of the plane—but you cannot walk around the passenger cabin and have a drink at the in-flight bar. Playing a videogame, in short, means performing a series of predetermined actions in which some or all of the actions performed occur repeatedly. The elementary form of intra-ludic seriality is thus not to be found in recurrent ludic structures (levels, lives, etc.) or features (enemies, power-ups, etc.) but in the set of operator actions that a player may execute.

Flappy Bird and *Flight Simulator X* represent two extremes in a continuum of intra-ludic seriality. Considering this continuum, one can characterize videogames by the way in which their gameplay implements the serialization of operator actions. A first, fundamental distinction could then be made between games that overtly exhibit serialized action and games that don't. A game may emphasize seriality not only by using recurrent structures and features but also by keeping the number of operator actions low, down to the bare minimum of one, as in *Flappy Bird* or other endless runners like *Canabalt* (Semi Secret Software 2009). Conversely, it can de-emphasize seriality not only by presenting the player with multiple, rich scenarios (levels, "worlds" etc.) but by offering her a wide range of options, as in *Flight Simulator X*'s dozens of cockpit controls. The two types of games described—let's call them *strongly* and *weakly serialized* for the sake of simplicity—correspond roughly to the contrary conceptual models of gameplay advocated by ludological and narratological positions. Strongly serialized titles like *Tetris* (Pajitnov et al. 1987) showcase the abstract, formal, rule-based nature of (video-)games theorized by ludologists. Weakly serialized titles like *Heavy Rain* (Quantic Dream 2010), on the other hand, display the narrative qualities of games discussed by narratologists.

Arcade Legacy

To be sure, the distinction between strongly and weakly serialized games is as problematic as the dichotomy between the ludological or narratological nature of videogames. Nevertheless, the distinction can serve as a basis for further investigation into the workings of intra-ludic seriality. Historically, videogames seem to have started out strongly serialized and have since gained an ever greater potential for weakly serialized gameplay. This is, of course, mainly due to the limited resources of the early hardware and software and the spectacular advances in integrated circuits during the last three decades. As arcade machines, PCs, and game consoles grew more and more powerful, the design of increasingly complex games became possible. That is how, in the genre of racing games, we have come from simple arcade racers like *Night Driver* (Michon 1976) to “real driving simulators” like *Gran Turismo 6* (Polyphony Digital 2013) and open world action-adventures like *Grand Theft Auto V* (Rockstar North 2013).

Indeed, arcade titles have been a major driver of strongly serialized gameplay. During the late 1970s and early 1980s—right before PCs and game consoles became mainstream and brought videogames into most private households—they were one of the principal sites for playing videogames. As the hardware used by arcade machines was relatively simple (typically one or more 8-bit microprocessors), so were the games. The decisive factor in gameplay design, though, was not the hardware but the socio-economic setting of the arcades, which forbid complex and elaborate forms of play. Arcade games not only had to appeal to a wide demographic; above all, they had to be easy-to-learn but hard-to-master so that they would generate a constant stream of coins being deposited into the machines. Steep learning curves (intuitive controls for a handful of operator actions) and steep increases in difficulty are therefore characteristic of arcade titles. Playing in the arcades is almost by definition a serial activity: “Game over. Insert coin to play again.”

Intra-ludic seriality permeated arcade games of the “golden age,” their recurrent graphics and sounds mirroring the repetitive gameplay on the machine’s surface. Multiple lives, linear progression through “screens” or levels, cyclical or infinite play[5] and scoring are structural hallmarks of arcade titles, exemplified by classics such as *Space Invaders* (Taito 1978), *Asteroids* (Atari 1979), *Galaxian* (Namco 1979), *Pac-Man* (Namco 1980), *Missile Command* (Atari 1980), and *Centipede* (Atari 1981). *Flappy Bird*’s design—the pixelated aesthetic, bare-bones game mechanism and extremely high difficulty—places it squarely within the tradition of these strongly serialized titles. It is the paradigmatic case of seriality in play. By reducing its gameplay to an absolute minimum, *Flappy Bird* reveals the fundamental dynamic embodied by arcade games: More often than not, playing a videogame means doing a series of actions *with the goal of being able to continue doing these actions for as long as possible.* The serialization of operator action aims at the (ideally endless) perpetuation of the series.[6] Game success is succession of operation[7]: You shoot rows of aliens so you can shoot even more aliens. You eat pellets (and avoid ghosts) in order to be given new pellets (and new ghosts). You destroy missile after missile only to be faced with yet another round of missiles. You flap the bird’s wings to keep it flying, flapping its wings again and again. Play is its own purpose.

Sticks and Buttons

With the commercial success of the PC and the third generation of game consoles in the 1980s came the decline of the arcade. The social setting and ever-increasing power of home platforms allowed videogames to evolve, and titles like *Elite* (Braben/Bell 1984), *The Legend of Zelda* (Nintendo R&D4 1986), *Maniac Mansion* (LucasFilm Games 1987), *Pirates!* (Microprose 1987), *SimCity* (Maxis 1989), and *Final Fantasy IV* (Square 1991) advanced the weakly serialized format. In recent years, however, the massive proliferation of smart phones, tablets, and portable media players has spawned a boom of “mobile games”: small titles such as *Flappy Bird* that embrace the qualities of early arcade games and mark a return to strongly serialized gameplay. Reality, though, is more complicated than is suggested by the picture of a historical cycle from basic forms of gameplay to more sophisticated ones and back to the simplicity of mobile games, from arcade machines to PCs and consoles to mobile devices, from entertainment centers to private homes to backpacks, purses, and pockets. For one thing, the range of operator action is heavily dependent on the genre of the game. Certain genres (prevalent in arcade machines), like shoot ’em ups, favor a more strongly serialized gameplay while others (prevalent on PCs and game consoles), like role-playing games, strive for greater freedom of choice. Text adventures are a case in point here, with some of the earliest titles such as *Zork I* (Infocom 1980) already demonstrating an impressive range of possible operator action owing to their clever text parsers. Secondly, while titles like *Heavy Rain* do provide a multitude of in-game actions, these actions are performed by the player through the repeated use of a few control elements, typically the buttons and sticks of a gamepad.

Naturally, this does not apply only to *Heavy Rain* and similar titles, but it is true for all kinds of videogames. And this is why, when speaking of operator actions, we have to distinguish carefully between *in-game actions* (e.g. the change in *Pac-Man*’s direction of movement through the maze displayed on the screen) and *interface actions* causing in-game actions (e.g. the push of the joystick mounted on the arcade cabinet) (cf. GamesCoop 2012, p. 55). For playing a game is always *playing at an interface,* whether the interface be a joystick, gamepad, mouse, or keyboard (or, more rarely, the player’s arms, hands, and feet tracked by motion sensing hard- and software). Trivial as the point may seem, it deserves closer examination.

In some cases, the same in-game action can be effected by way of different interface actions. For example, when running the MAME emulator software to play old arcade titles on a personal computer, the player can choose whether she wants to press keys on a keyboard or push a joystick to, say, direct Jumpman over a series of platforms. Far more common, though, is for the same interface action to cause a variety of different in-game actions depending on the game situation. In *Super Mario Bros.* (Nintendo EAD 1985), the gamepad’s A and B buttons control two separate actions each. Pressing the A button makes Mario (or Luigi) either jump or swim, depending on the situation, while the B button makes the character either run or throw a fireball. Because the link between interface and in-game action is determined by computer code, the possibilities are, in effect, endless, restricted only by the concrete means of input and the design of the game. So, in *Heavy Rain*, a few sticks, buttons, and triggers on the DualShock gamepad can control a myriad of in-game actions, from throwing a punch to determining the tone and topic of a conversation to using an asthma inhaler.

Hands and Fingers

The many different ways in which in-game and interface actions can be linked deserve a study of their own. Concerning the question of intra-ludic seriality at hand, we may content ourselves to state that restriction to a small number of possible in-game actions, standing in a strict one-to-one correspondence with respective interface actions, emphasizes the serial character of gameplay and produces strongly serialized games (whereas a larger number of in-game actions, in conjunction with one-to-many correspondences with the interface, de-emphasizes the serialization of operator action and leads to weakly serialized games). *Flappy Bird* is exemplary in this regard. One single interface action ("tap") causes one single in-game action ("flap").[8]

But whatever the relation between in-game action and interface action might be: in the end, all action in gaming comes down to a sequence of swift finger movements. To be sure, there are other kinds of interfaces, using speech and gesture recognition like the Microsoft Kinect and PlayStation Move systems, and these have gained prominence in recent years with titles like *Dance Central* (Harmonix 2010) and *Just Dance 3* (Ubisoft 2011). It is telling, though, that such interfaces are usually promoted and experienced as "extraordinary" controls—the exception rather than the norm. The norm is the use of hands and fingers on sticks and buttons. Whether it is *Space Invaders*, *Grand Theft Auto V*, or *Flappy Bird*—what we are really *doing* when playing these games is guiding our hands and fingers through a series of motions, pushing mice and sticks, clicking buttons and keys, swiping, pinching, and tapping trackpads and screens. And this fact, surprisingly, brings us back to our initial question about the nature of digital media.

Denson and Jahn-Sudmann treat *digital* seriality as a special case of popular seriality and suggest that the current digitization of popular media constitutes a "transformation, if not a radical break, in modern media history" (2013, p. 5). As computerized media enforce the "logic of the database" (Lev Manovich) on today's culture, established serial forms following the linear logic of the narrative give way to novel forms of seriality characterized by interactivity, flexibility, and synchronicity (cf. Denson and Jahn-Sudmann 2013, pp. 3-5). Denson's and Jahn-Sudmann's approach opens up new possibilities for the study of serialities in popular culture by calling attention to "the aesthetic forms and the cultural practices of serialization as they are articulated in and around interactive digital media" (pp. 10-11). Taking up the category of intra-ludic seriality, we have followed a path of investigation that has led us to consider the role of hands and fingers in playing videogames.

In the remainder of the paper, I would like to broaden the perspective on digital media a bit. I will try to sketch, very briefly, an alternative view of digital seriality in particular and of digitality in general—one that seeks to strategically shift the common notion of the word "digital" by bringing into focus the primary organ of digitality: the fingered hand.

Digital Seriality

Let us step back from the topic of videogames for a moment and start with this basic premise: *There is no seriality but digital seriality.*

A series is a row, succession, or sequence. The word derives from the Latin *serere* meaning “to link,” “to join,” or “to string together.”[9] Analytically, of course, what is linked or strung together in a series are separate parts. A series is a series in the strict sense of the word only when it is possible, at least in theory, to discern the individual parts of the sequence. (Otherwise, what you have is not a series but an undifferentiated, continuous wholeness, e.g. a line, curve, or wave.) For a series to exist, there need to be at least two discrete parts “in” it. Arranged in the right way, discrete parts act as elements of a digital system.

Note the distinction I draw between “discrete” and “digital.” Regarded in isolation, parts are simply discrete—discreteness meaning they are individual objects insofar as they can or could (if only hypothetically) exist independently of each other. It is when they are combined to constitute a system that discrete parts turn into *digital* elements. For it is the structural configuration of separate parts in a larger framework that renders each part of the framework a digital element. This is to say that “digital” is a relational or a functional term while “discrete” is typically used as an ontological term.[10] Collected in a glass jar, wooden beads are discrete objects; strung on wires in a frame, the same beads can act as digital elements of an abacus. Compared to the complex operations of the abacus, the functional systematicity of the series is a simple but powerful mechanism: The defined succession of elements permits easy and reliable shifting from any given position to every other by “moving” stepwise in one “direction” of the sequential order.

To be sure, not all digital systems are plain and simple series (as the abacus shows). But the series, being a very basic type of digitality, is a fundamental form of human culture. Seriality governs, among others, the two most important symbol systems of occidental culture: numerals and (alphabetic) letters. For more than 3,000 years, the standard sequence *a, b, g(c) …* has given order to the sets of written characters from the Ugaritic writing system to the Phoenician, Hebrew, Greek, and Latin scripts and the modern Western alphabets (cf. Naveh 1982, p. 11). The seriality of the alphabet allows, for example, for sorting and collation of information. Cornerstone techniques of data processing, alphabetic sorting and collating have made possible tools and aids such as dictionaries, encyclopedias, filing systems, library catalogues, and telephone directories. Our most recent and sophisticated tools for data processing, on the other hand, implement the seriality not of letters but of numbers: digital computers.

Numbers and Fingers

What is “digital” about digital computers? Commonly, the word is used to denote the fact that computers operate with instructions and on data that are expressed in numerical form. In science as well as in the humanities, numerical representation is taken to be one of the key principles of computers and digital media (cf. Manovich 2001, pp. 27-28). This notion of the “digital” fits well with the distinction between discrete and digital I have given above. On the level of both computer hardware and software, instructions and data are reduced to a minimal set of (usually two) discrete states: on and off in the case of the current in a transistor, one or zero in the case of a bit. Within the framework of circuits and programs, these discrete states operate as digital elements of logic gates and algorithms. Of course, integrated circuits and computer programs can be highly complex configurations with millions or even billions of elements.[11] Ultimately, though, electronic computation is nothing but a sequence of numerically expressed instructions and data to be executed and processed one after the other by a machine capable of carrying out the corresponding operations. On the diachronic axis and the most fundamental level of hard- and software, digital computing comes down to a series of (typically binary) numbers registering physically and logically in changes of state of transistors and bits.[12]

But the term “digital” not only designates the numerical representation of instructions and data that lies at the heart of electronic computation. The word also points us to the origin of number itself. Historically, the idea of abstract number developed from mastering the natural sequence *one, two, three …* The concept of the (infinite) sequence of natural numbers, in turn, most certainly goes back to the technique of finger counting (cf. Dantzig 1940, pp. 9-10). The human’s “number sense,” i.e. our ability to recognize the number of visually grouped objects without counting them, is extremely limited, typically not exceeding four objects (cf. Way n.d.). To correctly determine the number of larger quantities one must resort to counting. And in every culture that knows how to count (beyond the first few numbers, anyway), counting seems to be inextricably linked to the fingers of the hand, our *digits*. For in our fingers, humans have always close “at hand” a set of proxies for all things to be counted.

> It is to his *articulate ten fingers* that man owes his success in calculation. It is these fingers which have taught him to count and thus extend the scope of number indefinitely. Without this device the number technique of man could not have advanced far beyond the rudimentary number sense. And it is reasonable to conjecture that without our fingers the development of number, and consequently that of the exact sciences, to which we owe our material and intellectual progress, would have been hopelessly dwarfed. (p. 10)

In the first sentence of this passage, Dantzig makes a point (haplessly phrased and maybe even unintentionally) that is completely obvious, while its significance is easily overlooked: The fingers of the human hand make up a closed set of bodily “elements.” Together, they constitute a functional unit. But it is not only the fingers that, as Dantzig says, are “articulate.” So are, on a higher level, the hand and the body as a whole. The articulation of one hand into five individual yet interrelated fingers provides a set. Two articulated hands provide two sets or, to put it another way, a set of sets. With our two hands and two feet, we are given multiple series of digits. Elaborating Dantzig’s idea, Menninger claims that the multiple but finite seriality of fingers (and of toes) has led to the discovery of the single but infinite seriality of natural numbers:

> While matching [things] with pebbles and parts of the body will only result in a sequenced auxiliary set, the set of fingers and toes is by nature *articulated*: 5 fingers complete a hand, 10 two hands; 20 hands and feet! […] What is the

unforeseen implication of this articulation? It shows the way by which the counting series [Zählreihe] can proceed beyond the first words: Once ‘man’ is counted, the second counting begins in the same way as the first, then the third, the fourth and so on. In this way, the levels are stacked up. *Articulation creates the regular progress of the counting series* [Zählreihe]*. In this deep sense, articulation is a gift from nature.* (Menninger 1957, pp. 47-48; *my translation*)

Language often attests to the link between finger and number: The English word *digit*, for example, has preserved the double meaning of the Latin *digitus* for “finger” (or “toe”) and for the numerals 1 through 9; in Slavic languages, the words for “hand” (or “fist”) and “five” are closely related; also, many of the world’s tongues take ten, the number of fingers on both hands, as the base for numeration (meaning numbers greater then ten are expressed as compounds of individual words for ones, tens, hundreds and so on). Numbers, in short, are digital not only in that they are sequenced, discrete “objects” but also in that their concept originates in the ordered, discrete digits of the hand.

Digitality

What is “digital” about digital media? Every answer seems to point, like an index, at our fingers—the fingers that taught us to count and to devise of number as an abstract concept (cf. Dantzig 1940); the fingers that, since the beginning of graphism, have guided our instruments in writing and drawing (cf. Leroi-Gourhan 1964) and thus made possible mathematics (cf. Mersch 2005); the fingers that implemented mathematics in ever more technologically advanced machines and media, up to and including the computers of our day (cf. Kittler 1993); the fingers we use constantly to press the keys, buttons, and switches on almost any electric, electronic, or digital device from an alarm clock to an automated teller; the fingers that let us control the various interfaces of digital media, from the keyboard of our desktop computer to the touchscreen of our mobile phone; the fingers that help navigate Flappy Bird through pipe after pipe by a series of well-timed taps, counting the immemorial series of numbers: 1, 2, 3, …

Games Cited

Atari (1972) *Pong.* Atari (Arcade).

Atari (1979) *Asteroids.* Atari (Arcade).

Atari (1980) *Missile Command.* Atari (Arcade).

Atari (1981) *Centipede.* Atari (Arcade).

Braben, D., Bell, I. (1984) *Elite.* Acornsoft (BBC Micro, Acorn Electron).

Harmonix (2010) *Dance Central.* MTV Games/Microsoft Game Studios (Xbox 360).

Infocom (1980) *Zork I.* Infocom (Apple II, Commodore 64, IBM PC et al.).

LucasFilm Games (1987) *Maniac Mansion.* LucasFilm Games (Apple II, Commodore 64, IBM PC et al.)

Maxis (1989) *SimCity.* (Amiga, Commodore 64, IBM PC, Mac OS).

Michon, T. (1976) *Night Driver.* Atari/Micronetics (Arcade).

Microprose (1987) *Pirates!* MicroProse (Apple II, Commodore 64, IBM PC et al.).

Microsoft Game Studios (2006) *Flight Simulator X.* Microsoft Games Studios (Windows).

Namco (1979) *Galaxian.* Namco/Midway (Arcade).

Namco (1980) *Pac-Man.* Namco/Midway (Arcade).

Namco (1982) *Pole Position.* Namco/Atari (Arcade).

Nintendo EAD (1985) *Super Mario Bros.* Nintendo (NES/Famicom).

Nintendo R&D4 (1986) *The Legend of Zelda.* Nintendo (NES/Famicom).

Ngyuen, D. (2013) *Flappy Bird.* .GEARS Studios (Apple iOS, Android).

Pajitnov, A., Pavlovsky, D. and Gerasimov, V. (1987) *Tetris.* Spectrum Holobyte (MS-DOS).

Polyphony Digital (2013) *Gran Turismo 6.* Sony Computer Entertainment (PlayStation 3).

Psygnosis (1995) *Wipeout.* Psygnosis (PlayStation, MS-DOS).

Quantic Dream (2010) *Heavy Rain.* Sony Computer Entertainment (PlayStation 3).

Rockstar North (2013) *Grand Theft Auto V.* Rockstar Games (2013).

SeeThru.co.uk (2000) *Helicopter Game.* SeeThru.co.uk (Flash).

Semi Secret Software (2009) *Canabalt.* Semi Secret Software (Flash, Apple iOS, Android, PlayStation 3, PlayStation Portable, PlayStation Vita, Commodore 64).

Square (1991) *Final Fantasy IV.* Square (SNES).

Taito (1978) *Space Invaders.* Taito/Midway (Arcade).

UbiSoft (2011) *Just Dance 3.* Ubisoft (Wii, Xbox 360, PlayStation 3).

Williams Electronics (1980) *Defender.* Williams Electronics (Arcade).

References

Bogost, I. (2014) The Squalid Grace of Flappy Bird. Why Playing Stupid Games Staves off Existential Despair. *The Atlantic,* February 3 [Online]. Available at: http://www.theatlantic.com/technology/archive/2014/02/the-squalid-grace-of-flappy-bird/283526/ [Accessed: July 7, 2014].

Dantzig, T. (1940) *Number: The Language of Science.* London: George Allen & Unwin.

Denson, S. and Jahn-Sudmann, A. (2013). Digital Seriality: On the Serial Aesthetics and Practice of Digital Games. *eludamos. Journal for Computer Game Culture*, 7 (1), pp.1-32.

Galloway, A. R. (2006) *Gaming: Essays on Algorithmic Culture.* Minneapolis: University of Minnesota Press.

GamesCoop. (2012). *Theorien des Computerspiels.* Hamburg: Junius.

Juul, J. (2014) There Once Was A Game Called Flappy Bird. *The Ludologist*, Vol. 10 February [Online]. Available at: http://www.jesperjuul.net/ludologist/there-once-was-a-game-called-flappy-bird [Accessed: July 7, 2014].

Kittler, F. (1993) Geschichte der Kommunikationsmedien. In Huber, J. and Müller, A. M. (eds.) *Raum und Verfahren. Interventionen*. Basel, Frankfurt am Main: Stroemfeld/Roter Stern, pp.169-188.

Kushner, D. (2014) The Flight of the Birdman: Flappy Bird Creator Dong Nguyen Speaks Out. *Rolling Stone,* March 11 [Online]. Available at: http://www.rollingstone.com/culture/news/the-flight-of-the-birdman-flappy-bird-creator-dong-nguyen-speaks-out-20140311 [Accessed: July 7, 2014].

Leroi-Gourhan, A. (1964) *Le geste et la parole,* Vol. 1. Paris: Albin Michel.

Manovich, L. (2001) *The Language Of New Media.* Cambridge, MA; London: MIT Press.

McAfee (2014) McAfee Labs Threats Report June 2014 [Online]. Available at: http://www.mcafee.com/uk/resources/reports/rp-quarterly-threat-q1-2014.pdf [Accessed: 18 July 2014].

Menninger, K. (1957) *Zahlwort und Ziffer. Eine Kulturgeschichte der Zahl.* 2nd ed. Göttingen: Vandenhoeck und Ruprecht.

Mersch, D. (2005) Die Geburt der Mathematik aus der Struktur der Schrift. In Grube, G., Kogge, W. and Krämer, S. (eds.) *Schrift. Kulturtechnik zwischen Auge, Hand und Maschine*. München: Wilhelm Fink, pp.211-233.

Naveh, J. (1982) *Early History Of The Alphabet.* Leiden: E. J. Brill.

Pias, C. (ed.) (2003) *Cybernetics – Kybernetik: The Macy-Conferences 1946–1953*. Zürich, Berlin: Diaphanes.

Rigney, R. (2014) Inside the Brief Life and Untimely Death of Flappy Bird. *Wired*, February 12 [Online]. Available at: http://www.wired.com/2014/02/flappy-bird/ [Accessed: July 7, 2014].

Rollings, A. and Morris, D. (1999) *Game Architecture and Design.* Scottdale, AZ: Coriolis Group Books.

Schreier, J. (2014) Flappy Bird Is Making $50,000 A Day With Mario-Like Art [UPDATE 3]. *Kotaku*, February 6 [Online]. Available at: http://kotaku.com/flappy-bird-is-making-50-000-a-day-off-ripped-art-1517498140 [Accessed: July 4, 2014].

Stuart, K. (2014) Flappy Bird Is Dead – But Brilliant Mechanics Made It Fly. *The Guardian,* February 10 [Online]. Available at: http://www.theguardian.com/technology/2014/feb/10/flappy-bird-is-dead-but-brilliant-mechanics-made-it-fly [Accessed: July 7, 2014].

Tassi, P. (2014) Over Sixty 'Flappy Bird' Clones Hit Apple's App Store Every Single Day. *Forbes,* March 6 [Online]. Available at: http://www.forbes.com/sites/insertcoin/2014/03/06/over-sixty-flappy-bird-clones-hit-apples-app-store-every-single-day/ [Accessed: July 17, 2014].

Trumble, A. (2010) *The Finger: A Handbook.* New York: Farrar, Straus; Giroux.

Way, J. (n.d.) Number Sense Series: Developing Early Number Sense. NRICH – Enriching Mathematics [Online]. Available at: http://nrich.maths.org/2477/index?nomenu=1 [Accessed: July 17, 2014]

Notes

[1] The thumb, of course, also counts as a finger; cf. Trumble (2010, pp.207–221).

[2] Incidentally, the game's original title was *Flap Flap*.

[3] See, for example, Schreier (2014).

[4] Sid Meier famously defined a game as "a series of interesting choices" (Rollings and Morris 1999, p.38).

[5] While many arcade games theoretically offer an infinite play, some do have an end, if only because of programming errors like the famous "split-screen" bug in *Pac-Man*.

[6] There are exceptions, of course. In racing games like *Pole Position* (Namco 1982) or *Wipeout* (Psygnosis 1995), the goal is usually to finish a course as fast as possible. In many titles, though, achieving this goal qualifies the player to race on yet another course so that the principle of perpetual seriality applies even in this case.

7 In competitive games, like *Pong* (Atari 1972), the goal is to make one's own series of actions outlast, exceed, or break the opponent's series.

8 Even starting the game with the first tap on the screen makes Flappy Bird beat his wings.

9 The reconstructed Proto-Indo-European root of the Latin verb *serere* and noun *series* is *ser-.

10 John von Neumann, one of the fathers of digital computing, remarked that "the question regarding the continuous or digital character relates to the main functional traits or large, reasonably self-contained parts of the entire organ"; quoted in Pias (ed.) (2003, p.177).

11 In 2014, an Intel Core i7 CPU has more than a billion transistors, while the source code for Google's Chrome browser has about five million lines of code.

12 Note that in the "real world" of transistors there are no precise levels of voltage corresponding to the discrete values "on" and "off" or 1 and 0. Rather, there is a range of voltage and the actual level of voltage is compared to an internal threshold separating "high" from "low" signal levels which in turn represent 1 and 0 (or vice versa).