Manovich, Lev. "Understanding metamedia." *Software Takes Command*. New York: Bloomsbury, 2013. 107–158. *Bloomsbury Collections*. Web. 26 Apr. 2023. <<http://dx.doi.org/10.5040/9781472544988.ch-002>>.

Downloaded from Bloomsbury Collections, www.bloomsburycollections.com, 26 April 2023, 15:27 UTC.

Copyright © Lev Manovich 2013. You may share this work for non-commercial purposes only, provided you give attribution to the copyright holder and the publisher, and provide a link to the Creative Commons licence.

CHAPTER TWO

Understanding metamedia

“It [the electronic book] need not be treated as a simulated paper book since this is a new medium with new properties.”

Kay and Goldberg, “Personal Dynamic Media,” 1977

Today *Popular Science*, published by Bonnier and the largest science+tech magazine in the world, is launching Popular Science+ — the first magazine on the Mag+ platform, and you can get it on the iPad tomorrow...What amazes me is that you don’t feel like you’re using a website, or even that you’re using an e-reader on a new tablet device — which, technically, is what it is. *It feels like you’re reading a magazine.*” (emphasis is in the original.)

“Popular Science+,” posted on April 2, 2010.

<http://berglondon.com/blog/2010/04/02/popularscienceplus/>

The building blocks

I started putting this book together in 2007. Today is April 3, 2010, and I am editing this chapter. Today is also an important day in the history of media computing (which started exactly forty years ago with Ivan Sutherland’s Sketchpad)—Apple’s iPad tablet computer first went on sale in the US on this date. During the years I was writing and editing the book, many important developments made Alan Kay’s vision of a computer as the “first metamedium” more real—and at the same time more distant.

The dramatic cuts in the prices of laptops and the rise of cheap notebooks (and, in the years that followed, tablet computers), together with the continuing increase in the capacity and decrease in price of consumer electronics devices (digital cameras, video cameras, media players, monitors, storage, etc.) brought media computing to even more people. With the price of a notebook ten or twenty times less than the price of a large digital TV set, the 1990s' argument about the "digital divide" became less relevant. It became cheaper to create your own media than to consume professional TV programs via the industry's preferred mode of distribution. More students, designers, and artists learned Processing and other specialized programming and scripting languages specifically designed for their needs—which made software-driven art and media design more common. Perhaps most importantly, most mobile phones became "smart phones" supporting Internet connectivity, web browsing, email, photo and video capture, and a range of other media creation capabilities—as well as the new platforms for software development. For example, Apple's iPhone went on sale on June 29, 2007; on July 10 when the App Store opened, it already had 500 third-party applications. According to Apple's statistics, on March 20, 2010 the store had over 150,000 different applications and the total number of application downloads had reached 3 billion. In February 2012, the numbers of iOS apps reached 500,000 (not counting many more that Apple rejected), and the total number of downloads was already 25 billion.¹

At the same time, some of the same developments strengthened a different vision of media computing—a computer as a device for buying and consuming professional media, organizing personal media assets and using GUI applications for media creation and editing—but not imagining and creating "*not-yet-invented media*." Apple's first Mac computer, released in 1984, did not support writing new programs to take advantage of its media capacities. The adoption of the GUI interface for all PC applications by the software industry made computers much easier to use but in the same time took away any reason to learn programming. Around 2000, Apple's new paradigm of a computer as a "media hub" (or a "media center")—a platform for managing all personally created

¹ <http://www.apple.com/itunes/25-billion-app-countdown/> (March 5, 2012).

media—further erased the “computer” part of a PC. During the following decade, the gradual emergence of web-based distribution channels for commercial media, such as Apple iTunes Music Store (2003), internet television (in the US the first successful service was Hulu, publically launched on March 12, 2008), the e-book market (Random House and HarperCollins started selling their titles in digital form in 2002) and finally the Apple iBookstore (April 3, 2010), together with specialized media readers and players such as Amazon’s Kindle (November 2007) have added a new crucial part to this paradigm. (In the early 2010s we also got the Android app market, the Amazon app store, etc.) A computer became even more of a “universal media machine” than before—with the focus on consuming media created by others.

Thus, if in 1984 Apple’s first Apple computer was critiqued for its GUI applications and lack of programming tools for the users, in 2010 Apple’s iPad was critiqued for not including enough GUI tools for heavy duty media creation and editing—another step back from Kay’s Dynabook vision. The following quote from an iPad review by Walter S. Mossberg from the *Wall Street Journal* was typical of journalists’ reactions to the new device: “if you’re mainly a Web surfer, note-taker, social-networker and emailer, and a consumer of photos, videos, books, periodicals and music—this could be for you.”² The *New York Times*’ NYT’s David Pogue echoed this: “The iPad is not a laptop. It’s not nearly as good for creating stuff. On the other hand, it’s infinitely more convenient for consuming it—books, music, video, photos, Web, e-mail and so on.”³

Regardless of how much contemporary “universal media machines” fulfill or betray Alan Kay’s original vision, they are only possible because of it. Kay and others working at Xerox PARC built the first such machine by creating a number of media authoring and editing applications with a unified interface, as well as the technology to enable the machine’s users to extend its capacities. Starting with the concept Kay and Goldberg proposed in 1977, to sum up this work at PARC (the computer as “a *metamedium*” whose content is “a *wide range of already-existing and not-yet-invented media*”) in this chapter I will discuss how this

² <http://ptech.allthingsd.com/20100331/apple-ipad-review/> (April 3, 2010).

³ <http://gizmodo.com/5506824/first-ipad-reviews-are-in> (April 3, 2010).

concept redefines what media is. In other words, I will go deeper into the key question of this book: what exactly is media after software?

Approached from the point of view of media history, the computer metamedium contains two different types of media. The first type is *simulations of prior physical media extended with new properties*, such as “electronic paper.” The second type is a number of *new computational media that have no physical precedents*. Here are the examples of these “new media proper,” listed with names of the people and/or places usually credited as their inventors: hypertext and hypermedia (Ted Nelson); interactive navigable 3D spaces (Ivan Sutherland), interactive multimedia (Architecture Machine Group’s “Aspen Movie Map”).

This taxonomy is consistent with the definition of the computer metamedium given in the end of Kay and Goldberg’s article. But let us now ask a new question: what are the building blocks of these simulations of previously existing media and newly invented media? Actually we have already encountered these blocks in the preceding discussion but until now I have not explicitly pointed them out.

The building blocks used to make up the computer metamedium are *different types of media data* and the *techniques for generating, modifying, and viewing this data*. Currently, the most widely used data types are text, vector images and image sequences (vector animation), continuous tone images and sequences of such images (i.e., photographs and digital video), 3D models, geo-spatial data, and audio. I am sure that some readers would prefer a somewhat different list and I will not argue with them. What is important at this point for our discussion is to establish that we have multiple kinds of data rather than just one kind.

This points leads us to the next one: the techniques for data manipulation themselves can be divided into two types depending on which data types they can work on:

(A) The first type is *media creation, manipulation, and access techniques that are specific to particular types of data*.

In other words, these techniques can be used only on a particular data type (or a particular kind of “media content”). I am going to refer to these techniques as *media-specific* (the word “media” in this case really stands for

“data type”). For example, the technique of geometrical constraint satisfaction invented by Sutherland can work on graphical data defined by points and lines. However, it would be meaningless to apply this technique to text. Another example: today image editing programs usually include various filters such as “blur” and “sharpen” which can operate on continuous tone images. But normally we would not be able to blur or sharpen a 3D model. Similarly, it would be as meaningless to try to “extrude” a text or “interpolate” it as to define a number of columns for an image or a sound composition.

Some of these data manipulation techniques appear to have no historical precedents in physical media—the technique of geometric constraint satisfaction is a case in point. Another example of such new technique is evolutionary algorithms commonly used to generate still images, animations, and 3D forms. Other media-specific techniques do refer to prior physical tools or machines—for instance, brushes in image editing applications, a zoom command in graphics software, or a trim command in video editing software. In other words, the same division between simulations and “properly new” media also applies to the individual techniques that make up the “computer metamedium.”

- (B) The second type is *new software techniques that can work with digital data in general (i.e. they are not media-specific)*. The examples are “view control,” hyperlinking, sort, search, network protocols such as HTTP, and various data analysis techniques from the fields of Artificial Intelligence, Machine Learning, Knowledge Discovery, and other sub-fields of computer science. (In fact, large parts of computer science, information science and computer engineering science are about these techniques—since they focus on designing algorithms for processing information in general.) These techniques are general ways of manipulating data regardless of what this data encodes (i.e. pixel values, text characters, sounds, etc.). I will be referring to these techniques as *media-independent*. For instance, as we saw, Engelbart’s “view control”—the idea that the same

information can be displayed in many different ways—is now implemented in most media editors and therefore works with images, 3D models, video files, animation projects, graphic designs, and sound compositions. “View control” has also become part of the modern OS (operating systems such as Mac OS X, Microsoft Windows, or Google Chrome OS). We use view control daily when we change the files “view” between “icons,” “list,” and “columns” (these are names used in Mac OS X; other OS may use different names to refer to the same views). General media-independent techniques also include interface commands such as cut, copy, and paste. For instance, you can select a file name in a directory, a group of pixels in an image, or a set of polygons in a 3D model, and then cut, copy, and paste these selected objects.

OK: we now have two different ways of “dividing up” the computer metamedium. If we want to continue using the concept of a “medium,” we will say that a computer simulates prior mediums and allows for the definition of new ones. Alternatively, we can think of the computer metamedium as a collection of data types, media-specific techniques that can only operate on particular types, and media-independent techniques that can work on any data. Each of the mediums contained in the computer metamedium is made from some of these building blocks. For example, the elements of the “navigable 3D space” medium are 3D models plus techniques for representing them in perspective, texture mapping, simulating effects of various types of lights on their surface, casting shadows, and so on. In another example, the elements of “digital photography” are continuous tone images captured by lens-based sensors plus a variety of techniques for manipulating these images: changing contrast and saturation, scaling, compositing, etc.

A note about the distinction between media-specific and media-independent techniques: it works in theory. In practice, however, it is often hard to say in what category a particular technique or a medium should be placed. For instance, is Sketchpad’s ability to work at any of 2000 levels of magnification an extension of techniques which existed previously (moving one’s body closer to the drawing board, using a magnifying lens) or is it something really new? Or what about 3D navigable space, which I have

used as an example of a new medium only made possible by computers (tracing it to Sutherland's first Virtual Reality system of 1966)? Is it new—or is it an extension of a physical medium of architecture, which allows a human being to walk around the built structures?

The boundaries between “simulated media” and “new media,” or between “media-specific” and “media-independent” techniques should not be thought of as solid walls. Rather than thinking of them as rigidly defined categories, let us imagine them as coordinates of the map of the computer metamedium. Like any first sketch, no matter how imprecise, this map is useful because now we have something to modify as we go forward.

Media-independent vs. media-specific techniques

Having drawn our first map of the computer metamedium, let us now examine it to see if it can reveal something which we did not notice so far. We see a number of mediums, old and new—and this certainly fits with our common understanding of media history. (For example, take a look at the table of contents of McLuhan's *Understanding Media* and you will find two dozen chapters each devoted to a particular medium—which for McLuhan range from writing and roads to cars and TV.) We also see various media-specific techniques and this again is something we are familiar with: think of editing techniques in cinema, defining a contour in painting, creating rhyme in poetry, or shaping a narrative out of chronological story events in literature. But one area of the map does look new and different in relation to previous cultural history. This is the area that contains “media-independent techniques.” What are these techniques, and how can they work across media, i.e. on different types of data? (“Design across media” was a phrase used by Adobe in marketing an early version of its Creative Suite of media authoring applications.)

I am going to argue that “media independence” does not just happen by itself. For a technique to work with various data types, programmers have to implement a different method for each data type. Thus, *media-independent techniques are general concepts*

translated into algorithms, which can operate on particular data types. Let us look at some examples.

Consider the omnipresent *cut* and *paste*. The algorithm to select a word in a text document is different from the algorithm to select a curve in a vector drawing, or the algorithm to select a part of a continuous tone (i.e. raster) image. In other words, “cut and paste” is a general concept that is implemented differently in different media software depending on which data type this software is designed to handle. (In Larry Tesler’s original implementation of the universal commands concept done at PARC in 1974–5, it only worked for text editing.) Although cut, copy, paste, and a number of similar “universal commands” are available in all contemporary GUI applications for desktop computers (but not necessarily in mobile phone apps), what they actually do and how they do it is different from application to application.

Search operates in the same way. The algorithm to search for a particular phrase in a text document is different than the algorithm that searches for a particular face in a photo or a video clip. (I am talking here about “content-based search,” i.e. the type of search which looks for information inside actual images, as opposed to only searching image titles and other metadata the way image search engines such as Google Image Search were doing it in the 2000s.) However, despite these differences the general concept of search is the same: locating any elements of a single media object—or any media objects in a larger set—to match particular user-defined criteria. Thus we can ask the web browser to locate all instances of a particular word in a current web page; we can ask a web search engine to locate all web pages which contain a set of keywords; and we can ask a content-based image search engine to find all images that are similar in composition to an image we provided.

Because of the popularity of the search paradigm on the web, we now assume that in principle we can—or will be able to in the future—search any media. In reality it is much easier to search data that has a modular organization—such as text or 3D models—than media that does not have it, such as continuous-tone images, video, or audio. But for the users these differences are not important—as far as they are concerned, all types of media content acquire a new common property that can be called *searchability*.

Similarly, in the mid-2000s photo and video media started

to acquire another property of *findability*. (I am borrowing this term from 2005 book by Peter Morville, *Ambient Findability: What We Find Changes Who We Become*⁴). The appearance of consumer GPS-enabled media capture devices and the addition of geo-tagging, geo-search, and mapping services to media sharing sites such as Flickr (added in 2006) and media management applications such as iPhoto (added in 2009) gradually made media “location aware.”

Another example of a general concept that, through the efforts by many people, was gradually made to work with different media types—and thus became a “media-independent technique”—is *information visualization* (often abbreviated as *infovis*.) The name “infovis” already suggests that it is not a media-specific technique—rather it is a very general method that potentially can be applied to any data. The name implies that we can potentially take anything—numbers, text, network, sound, video, etc.—and map it into image to reveal patterns and relationships in the data. (A parallel to information visualization is *data sonification*, which renders data as sound).

However, it took decades to invent techniques to turn this potential into reality. In the 1980s the emerging field of scientific visualization focused on 3D visualization of numerical data. In the second part of the 1990s the growing graphics capabilities of PCs made possible for larger numbers of people to experiment with visualization—which led to the development of techniques to visualize media. The first successful visualizations of large bodies of text appeared around 1998 (*Rethinking the Book* by David Small, 1998; *Valence* by Ben Fry, 1999; *TextArc* by W. Bradford Paley, 2002⁵); visualizations of musical structures in 2001 (*The Shape of Song* by Martin Wattenberg); and visualization of a feature film in 2000 (*The Top Grossing Film of All Time, 1 x 1* by Jason Salovan).

Information visualization is a particularly interesting example of a new “media-independent technique” because of the variety of the algorithms and strategies for representing data visually.

⁴ Peter Morville. *Ambient Findability: What We Find Changes Who We Become*. O'Reilly Media, Inc., 2005.

⁵ W. Bradford Paley, *TextArc*, 2002, <http://www.textarc.org/>; Ben Fry, *Valence*, 1999, <http://benfry.com/valence/>; David Small, *Rethinking the Book*, PhD thesis, 1999, <http://acg.media.mit.edu/projects/thesis/DSThesis.pdf>

Wind Map, a real-time dynamic visualization of the wind currents over the USA. Fernanda Viégas and Martin Wattenberg, 2012.

For example, Martin Wattenberg whose work, in his own words, “focuses on visual explorations of culturally significant data,”⁶ created visualizations of a history of net art, the music compositions of Bach, Philip Glass and other composers, the thought process of a computer chess-playing program, and the history of Wikipedia pages, among other projects. In each case he had to decide which dimensions of the data to choose and how to translate them in a visual form. But despite the differences, we recognize all these projects as information visualizations. They are all realizations of the same general concept—selecting some dimensions of the data and representing them visually through the relations of graphic elements.⁷ They also all rely on the same fundamental capacities of software to manipulate numerical data and to map it from one form to another. Finally, they all can be also understood as the application of the new computer media of computer graphics—generation of images from numerical data. (Think of an affinity between a 3D computer model based on a 3D scan of a face, and a vector visualization of, for instance, the face’s position over time, based on the data extracted from a video.)

As the result of infovis work by Wattenberg and other people over the course of last two decades many types of data acquired a new common property—their structure can be visualized. This new property of media is distributed across various applications, software libraries, art and design projects, research papers, and prototypes. Today some visualization tools are included in media editing software—for instance, media editors such as Photoshop can display a histogram of an image, Final Cut and other professional video editing software can visualize the color content of a video clip, and many media players including iTunes offer a music visualization feature. Google Trends visualizes search patterns; YouTube and Flickr visualize viewing stats for video and photos. Going through the thousands of infovis projects collected on infosthetics.com, visualcomplexity.com, and other blogs about visualizations, we find a variety of experiments in visualization of media such as songs, poems and novels and every possible kind

⁶ <http://www.bewitched.com/about.html> (July 23, 2006).

⁷ For a detailed discussion of infovis, most general principles and new developments, see my article “What is Visualization?” (2010), *Visual Studies* 26, no. 1 (March 2011).

of data—from the artist's son's, daughter's and cat's movements in their living room over a period of an hour (*1hr in front of the TV* by umblebee, 2008) to a citation network in science journals (*Eigenfactor.org* by Moritz Stefaner, 2009).⁸ We can also find such projects in art exhibitions such as MOMA's 2008 *Design and Elastic Mind*,⁹ SIGGRAPH 2009 *Info-Aesthetics*,¹⁰ and MOMA's 2011 *Talk to Me*.

Visualization, searchability, findability—these and many other new “media-independent techniques” (i.e. concepts implemented to work across many data types) clearly stand out in the map of the computer metamedium we have drawn because they go against our habitual understanding of media as plural (i.e. as consisting of a number of separate mediums). If we can use the same techniques across different media types, what happens to these distinctions between mediums?

The idea that all artworks fall into a number of distinct mediums each with its own distinct techniques and representational devices was central to modern art and aesthetics. In his 1766 *Laokoon oder Über die Grenzen der Malerei und Poesie* (*Laocoon: An Essay on the Limits of Painting and Poetry*) German philosopher Gotthold Ephraim Lessing argued for the radical difference between poetry and painting since one is “extended” in time and the other is in space. The idea reached its extreme in the first two decades of the twentieth century when modernist artists focused their energy on discovering a unique language of each artistic medium. The following statement made in 1924 by Jean Epstein, a French avant-garde filmmaker and theoretician, is typical of modernist rhetoric of purity; countless statements like it appeared on the pages of avant-garde publications of the time:

For every art builds its forbidden city, its own exclusive domain, autonomous, specific and hostile to anything that does not belong. Astonishing to relate, literature must first and foremost be literary; the theater, theatrical; painting, pictorial; and the cinema, cinematic. Painting today is freeing itself from many of

⁸ <http://well-formed.eigenfactor.org/>; http://www.flickr.com/photos/the_bumblebee/2229041742/in/pool-datavisualization

⁹ <http://www.moma.org/interactives/exhibitions/2008/elasticmind/>

¹⁰ http://www.siggraph.org/s2009/galleries_experiences/information_aesthetics/

its representational and narrative concerns... The cinema must seek to become, gradually and in the end uniquely, cinematic; to employ, in other words, only photogenic elements.¹¹

In relation to painting, the doctrine of media purity reaches its extreme expression in the famous argument of Clement Greenberg that “Because flatness was the only condition painting shared with no other art, Modernist painting oriented itself to flatness as it did to nothing else.”¹² (Note that Greenberg did not advocate this position as a justification for the abstract art of his contemporaries; he only offered this as a historical analysis of earlier modernism.)

Greenberg wrote: “It was the stressing of the ineluctable flatness of the surface that remained, however, more fundamental than anything else to the processes by which pictorial art criticized and defined itself under Modernism. For flatness alone was unique and exclusive to pictorial art.”

Only after the 1960s when installation—a new art form based on the idea of mixing different media and materials—gradually became popular and accepted in the art world, did the obsession with media-specificity lose its importance.

However, even during its dominance the principle of media-specificity was always counterbalanced by its opposite. Throughout the modern period we also find “local”—i.e. specific to particular historical moments and artistic schools—attempts to formulate aesthetic principles that can relate different mediums to each other. Consider for instance the considerable efforts spent by many modernist artists to establish parallels between musical and visual compositions. This work was often associated with the ideas of synesthesia and *Gesamtkunstwerk*; it included theories, practical compositions, and technologies such as color organs constructed by Scriabin, the Whitneys (who went on to create the first computer animations) and many other artists and musicians.

While they did not explicitly theorize cross-media aesthetics to the same degree, modernist artistic paradigms—classicism,

¹¹ Jean Epstein, “On Certain Characteristics of *Photogénie*,” in *French Film Theory and Criticism*, vol. 1: 1907–29, ed. Richard Abel (Princeton: University of Princeton Press, 1988).

¹² Clement Greenberg, “Modern Painting,” *Forum Lectures* (Washington, DC: Voice of America: 1960), <http://www.sharecom.ca/greenberg/modernism.html>

romanticism, naturalism, impressionism, socialist realism, suprematism, cubism, surrealism, and so on—can be also understood as the systems which gave “common properties” to works in various media. Thus, the novels of Émile Zola and paintings by Manet were aligned in their uncompromising, “naturalist” depiction of ordinary people; Constructivist paintings, graphics, industrial design, theatre design, architecture and fashion shared the aesthetics of “expressed structure” (visually emphasizing composition structure by exaggerating it); and De Stijl aesthetics of non-intersecting rectangular forms in primary colors were applied to painting, furniture, architecture, and typography.

What is the difference between such earlier artistic work on establishing media correspondences and the software techniques that work across different media? Clearly, the artistic systems and media authoring, editing and interaction techniques available in media software operate on different levels. The former are responsible for the content and style of the works to be generated—i.e. what is going to be created in the first place. The latter are used not only to create but also to interact with what already was generated previously—for instance, blogs, photos, or videos on the web created by others.

Put differently, the efforts by modern artists to create parallels between mediums were prescriptive and speculative.¹³ “Common media properties” would only apply to selected bodies of artistic work created by particular artists or groups. In contrast, software imposes common media “properties” on any media it is applied to. Thus, software also shapes our understanding of what media is in general. For example, web applications and services include methods for navigating, reading, listening or viewing media objects, attaching additional information to them (comments, tags, geo-tagging) or finding them in a larger set (i.e. search engines and search commands). This applies to all videos, images, text pages, text documents, maps, etc. In other words, we can say that media

¹³ By speculative here I mean that in many cases the proposed aesthetic systems were not fully realized in practice. For example, no purely suprematist architecture designed by movement leader Kasimir Malevich was ever built; the same goes for the futurist architecture of Antonio Sant’Elia as presented in his drawings for *La Città Nuova*, 1912–14.

software “interprets” any media it touches and its “interpretations” always include certain statements.

Of course, “media-independent” aesthetic systems proposed by modernists were not only generative (the creation of new works) but also interpretive. That is, modernist artists and theorists often tried to change audiences’ understanding of past and contemporary art, usually in a critical and negative way (each new movement wanted to discredit its predecessors and competitors). However, since their programs were theories rather than software, they had no direct material effect on users’ interaction with the artistic works, including those created in the past. In contrast, software techniques affect our understanding of media through the operations they make available to us for creating, editing, interacting with, and sharing media artifacts.

Additionally, modern artistic and aesthetic paradigms in practice would only be realized in two, three, or maybe four mediums—but not all. Naturalism can be found in literature and visual arts but not in architecture, and Constructivism did not spread to music or literature. However cut, copy, paste and find commands are found in all media applications; any media object can be geo-tagged; the view control principle is implemented to work with all media types. To cite more examples, recall my discussion of how all media acquire new properties such as searchability and findability. Any text can be searched regardless of whether it is something you wrote or a classical novel downloaded from Project Gutenberg; similarly, part of an image can be cut and pasted into another image regardless of what these images are. In short: media software affects all media content equally, regardless of its aesthetics, semantics, authorship, and historical origin.

To summarize this discussion: in contrast to modern artistic programs to create different media that share the same principles, software media-independent techniques are ubiquitous and “universalist.” For instance, cut and paste are built into all media editing software—from specialized professional applications to consumer software included on every new media device sold. Further, these techniques can be applied to any media work regardless of its aesthetics and authorship—i.e. whether it was made by the person who is currently applying these operations or by somebody else. In fact, the technical ability to sample media work by others has become the basis of the key aesthetics of our time—remixing.

Of course, not all media applications and devices make all these techniques equally available—usually for commercial and/or copyright reasons. For example, at present the Google Books reader does not allow you to select and copy the text from book pages. Thus, while our analysis applies to conceptual and technical principles of software and their cultural implications, we need to keep in mind that in practice these principles are overwritten by commercial structures. However, even the most restrictive software still includes some basic media operations. By being present in all software designed to work with different media types, these operations establish a shared understanding of media today—experienced by the users as “common media principles” of all content.

To conclude this discussion of software techniques that work across different media types, recall my earlier statement that computerization of media does not collapse the difference between mediums—but it does bring them closer together in various ways. (I have already provided a number of examples of this in the beginning of my analysis of Kay and Goldberg’s *Personal Dynamic Media* article). Now I can name one of the key developments responsible for this “media attraction”—common software techniques which can operate across different media types. If we recall again Kay and Goldberg’s formulation that the computer metamedium includes a variety of already-existing and new media, this statement can be paraphrased as follows: *within the computer metamedium, all previously existing and newly invented mediums share some common properties—i.e. they rely on a set of common software techniques for data management, authoring, and communication.*

It is hard to over-estimate the historical importance of the development of these cross-media techniques. Humans have always used some general strategies to organize their cultural representations, experiences and actions—for example, narrative, symmetry, rhythm, the repetitive structures (think of ornament), use of complementary colors, and a few others. Clearly these strategies were very important for human perception, cognition, and memory—and that is why we find them in every culture and every medium, from poetry to architecture, music and poetry. However, these strategies were normally not embedded into any technological tools or materials—they were in the minds and bodies of artisans who were communicating them from generation to

generation. Modern media for representation and communication bring us to a new stage. They often do embody certain techniques that apply to any media which can be generated or captured with them (for instance, one-point linear perspective imposed by lens-based capture technologies such as photography, film and analog and digital video). However these techniques would apply only to particular media types. Against these historical developments, the innovation of media software clearly stands. They bring a new set of techniques which are implemented to work across all media. *Searchability*, *findability*, *linkability*, multimedia messaging and sharing, editing, view control, zoom and other “media-independent” techniques are viruses that infect everything software touches—and therefore in their importance they can be compared to the basic organizing principles for media and artifacts which were used for thousands of years.

Inside Photoshop

Contemporary media is experienced, created, edited, remixed, organized, and shared with software. This software includes stand-alone professional media design and management applications such as Photoshop, Illustrator, Dreamweaver, Final Cut, After Effects, Aperture, and Maya; consumer-level apps such as iPhoto, iMovie, or Picasa; tools for sharing, commenting, and editing provided by social media sites such as Facebook, YouTube, Vimeo, and Photobucket, and the numerous media viewing, editing, and sharing apps available on mobile platforms. To understand media today we need to *understand media software*—its genealogy (where it comes from), its anatomy (interfaces and operations), and its practical and theoretical effects. How does media authoring software shape the media being created, making some design choices seem natural and easy to execute, while hiding other design possibilities? How does media viewing/managing/remixing software affect our experience of media and the actions we perform on it? How does software change what “media” is conceptually?

This section continues investigating these general questions that drive this book via the analysis of a software application that became synonymous with “digital media”—Adobe Photoshop.

Like other professional programs for media authoring and editing, Photoshop's menus contain many dozens of separate commands. If we consider that almost all the commands contain multiple options that allow each command to do a number of different things, the complete number runs into the thousands.

This multiplicity of operations offered in contemporary application software creates a challenge for Software Studies. If we are to understand how software applications participate in shaping our worlds and our imaginations (what people imagine they can do with software), we need some way of sorting all these operations into fewer categories so we can start building a theory of application software. This cannot be achieved by simply following the top menu categories offered by applications. (For example, Photoshop CS4's top menu includes File, Edit, Layer, Select, Filter, 3D, View, Window, and Help.) Since most applications include their own unique categories, our combined list will be too large. So we need to use a more general system.

The provisional map of the computer metamedium that we developed in previous sections provides one such possible system. In this section we will test the usefulness of this map by analyzing a subset of Photoshop's commands which, in a certain sense, stand-in for this application in our cultural imagination: Filters. We will also discuss another key feature of Photoshop—Layers.

Our map organizes software techniques for working with media, using two schemes. The first scheme divides these techniques into two types depending on which data types they can work on: 1) media creation, manipulation, and access techniques that are specific to particular types of data; 2) new software techniques that can work with digital data in general (i.e. they are not specific to particular types of data). My second scheme also divides software techniques for working with media data into two types, but it does this in a different way. What matters here are the relations between software techniques and pre-digital media technologies. In this taxonomy, some techniques are simulations of pre-digital media techniques augmented with new properties and functions; other techniques do not have any obvious equivalents in previous physical or electronic media.

While for a media historian the second scheme is quite meaningful, what about users who are "digital natives"? These software users may never have directly used any other media besides tablets or

5.5

6.0

7.0

laptops, or mobile media devices (mobile phones, cameras, music players); and they are also likely to be unfamiliar with the details of twentieth-century cel animation, film editing equipment, or any other pre-digital media technology. Does this mean that the distinction between software simulations of previously existing media tools and new “born digital” media techniques has no meaning for digital natives but only matters for historians of media such as myself?

I think that while the semantics of this distinction (i.e. the reference to previous technologies and practices) may not be meaningful to digital natives, the distinction itself is something these users experience in practice. To understand why this is the case, let us ask if all “born digital” media techniques available in media authoring software applications may have something in common—besides the fact that they did not exist before software.

One of the key uses of digital computers from the start was *automation*. As long as a process can be defined as a finite set of simple steps (i.e. as an algorithm), a computer can be programmed to execute these steps without human input. In the case of application software, the execution of any command involves “low-level” automation (since a computer automatically executes a sequence of steps of the algorithm behind the command). However, what it is important from the user’s point of view is the level of automation being offered in the command’s interface.

Many software techniques that simulate physical tools share a fundamental property with these tools: they require a user to control them “manually.” The user has to micro-manage the tool, so to speak, directing it step-by-step to produce the desired effect. For instance, you have to explicitly move the cursor in a desired pattern to produce a particular brushstroke using a brush tool; you also have to explicitly type every letter on a keyboard to produce a desired sentence. In contrast, many of the techniques that do not simulate anything that existed previously—at least, not in any obvious way—offer higher-level automation of creative processes. Rather than controlling every detail, a user specifies parameters and controls and sets the tool in motion. All generative (also called “procedural”) techniques available in media software fall into this category. For example, rather than having to create a rectangular grid made of thousands of lines by hand, a user can specify the width and the height of the grid and the size of one cell, and the

program will generate the desired result. Another example of this higher-level automation is interpolation of key values offered by animation software. In a twentieth-century animation production, a key animator drew key frames which were then forwarded to human in-betweeners who created all the frames *in between* the key frames. Animation software automates the process of creating in-between drawings by automatically interpolating the values between the key frames.

Thus, although users may not care that one software tool does something that was not possible before digital computers while another tool simulates previous physical or electronic media, the distinction itself between the two types is something users experience in practice. The tools that belong to the first type showcase the ability of computers to automate processes; the tools that belong to the second type use invisible low-level automation behind the scenes while requiring users to direct them manually.

Filter › stylize › wind

Having established two sets of categories for software techniques (media-independent *vs.* media-specific; simulation of the old *vs.* new), let us now test them against the Photoshop commands. Before starting, however, it is important to note once again that the two proposed schemes are intended to serve only as provisional categories. They provide one possible set of directions—an equivalent of North, South, West and East for a map where we can locate multiple operations of media design software. Like any first sketch, no matter how imprecise, this map is useful because now we have something to modify as we go forward. Our goal is not to try to fit everything we will look at into the categories of this initial map, but rather to discover its limitations as quickly as we can, and make modifications.

We will start with Photoshop filters—i.e. the set of commands that appear under the Filter menu. (Note that a large proportion of Photoshop filters are not unique to this program but are also available in other professional image editing, video editing and animation software—sometimes under different names. To avoid any possible misunderstanding, I will be referring to the Photoshop versions of these commands as implemented in Photoshop CS4,

with their particular options and controls as defined in this software release.¹⁴)

The first thing that is easy to notice is that the names of many Photoshop filters refer to the techniques for image manipulation and creation and materials that were available before the development of media application software in the 1990s—painting, drawing and sketching, photography, glass, neon, photocopying. Each filter is given a set of explicit options that can be controlled with interactive sliders and/or by directly entering desired numbers. These controls not only offer many options but also often allow you to set filter's properties numerically by choosing a precise value from a range.

This is a good example of my earlier point that simulations of prior physical media augment them with new properties. In this case, the new property is the explicit filter controls. For example, the Palette Knife filter offers three options: Stroke Size, Stroke Detail, and Softness. Stroke Size can take values between 1 and 50; the other two options have similarly large ranges. (At the same time, it is important to note that expert users of many physical tools such as a paintbrush can also achieve many effects not possible in its software simulation. Thus, software simulations should not be thought of simply as improvements over previous media technologies.)

While some of these filters can be directly traced to previous physical and mechanical media such as oil painting and photography, others make a reference to actions or phenomena in the physical world that at first appear to have nothing to do with media. For instance, the Extrude filter generates a 3D set of blocks or pyramids and paints image parts on their faces, while the Wave filter creates the effect of ripples on the surface of an image.

However, if we examine any of these filters in detail, we realize that things are not so simple. Let us take the Wind filter (located under the Stylize submenu) as an example. This is how Photoshop CS4's built-in Help describes this filter: "Places tiny horizontal lines in the image to create a windblown effect. Methods include Wind; Blast, for a more dramatic wind effect; and Stagger, which offsets the lines in the image." We are all familiar with the visual effects

¹⁴ For a history of Photoshop version releases, see http://en.wikipedia.org/wiki/Adobe_Photoshop_release_history

The effects of Photoshop Wind filter (version CS5.1). Left to right: original shape, no filter; the filter with “wind” option applied; the filter with “blast” option applied.

of a strong wind on a physical environment (for instance, blowing through a tree or a field of grass)—but before you encountered this filter, you probably never imagined that you can “wind” an image. Shall we understand the name of this filter as a metaphor? Or perhaps, we can think of it as an example of a conceptual “blend” (which is how, according to Conceptual Blending theory, many concepts in natural languages get formed¹⁵): “wind” plus “image” results in a new concept actualized in the operations of the Wind filter.

The situation is further complicated by the fact that the results of applying the Wind filter to an image look pretty different compared to what the actual wind does to a tree or a field of grass. However, they do look rather similar to a photograph of a real windy scene taken with a long exposure. Therefore, we can think of the name “Wind” both as a metaphor—to help us imagine what a particular algorithmic transformation does to an image—and as a simulation of a particular photographic technique (long exposure). In short, although its name points to the physical world, its actual operations may also refer to a pre-digital media technology.

Are there “born digital” filters?

Let us continue the exploration of Photoshop filters. The great majority of the filters make references to previous physical media

¹⁵ See Mark Turner and Gilles Fauconnier, *The Way We Think. Conceptual Blending and the Mind’s Hidden Complexities* (New York: Basic Books 2002).

or our experiences in the physical world—at least in terms of how they are named. Only a few do not. These filters include High-pass, Median, Reduce Noise, Sharpen, and Equalize. Are these filters “born digital”? In other words, did we finally get to pure examples of “new” media techniques? The answer is no. As it turns out, all these filters are also software simulations that refer to things that already existed before digital computers.

Although they represent a small subset of Photoshop’s extensive filter collection, these filters are central to all electronics, telecommunication, and IT technologies. Moreover, they are not unique to processing digital images but can be used on any kind of data—sounds, television transmission, data captured by an environmental sensor, data captured by a medical imaging devices, etc.

In their Photoshop implementation, these filters work on continuous-tone images, but since they can be also applied to sound and other types of signals, they actually belong to our “media-independent” category of software techniques. In other words, they are general techniques developed first in engineering and later also in computer science for signal and information processing. The application of these techniques to images forms the part of the field of image processing defined as “any form of information processing for which the input is an image, such as photographs or frames of video.”¹⁶ This conceptual relationship between “information processing” and “image processing” exemplifies one of the key points of this book—in software culture, “digital media” is a particular subset of the larger category “information.” (Thus, the operations commonly used with media form a subset of the larger category “data processing.”)

Like these filters, *many of the “new” techniques for media creation, editing, and analysis implemented in software applications were not developed specifically to work with media data.* Rather, they were created for signal and information processing in general—and then were either directly carried over to, or adapted to work with media. (Thus, development of software brings different media types closer together because the same techniques can be used on all of them. At the same time, “media” now share

¹⁶ *Ibid.*

a relationship with all other information types, be they financial data, patient records, results of a scientific experiments, etc.)

This is one of the most important theoretical dimensions in the shift from physical and mechanical media technologies to electronic media and then digital software. Previously, physical and mechanical media tools were used to create content which was directly accessible to human senses (with some notable exceptions like Morse code)—and therefore the possibilities of each tool and material were driven by what was meaningful to a particular human sense. A paintbrush could create brushstrokes that had color, thickness, and shape—properties directly speaking to human vision and touch. Similarly, the settings of photographic camera controls affected the sharpness and contrast of captured photos—characteristics meaningful to human vision. A different way to express this is to say that the “message” was not encoded in any way; it was created, stored, and accessed in its native form. So if we were to redraw the famous diagram of a communication system by Claude Shannon (1948)¹⁷ for the pre-electronics era, we would have to delete the encoding and decoding stages.

Successive media technologies based on electronics (such as the telegraph, telephone, radio, television), and digital computers employ the coding of messages or “content.” And this, in turn, makes possible the idea of *information*—a disembodied, abstract and universal dimension of any message separate from its content. Rather than operating on sounds, images, video, or texts directly, electronic and digital devices operate on the continuous electronic signals or discrete numerical data. This allows for the definition of various operations that work on any signal or *any set of numbers*—regardless of what this signal or numbers may represent (images, video, student records, financial data, etc.). Examples of such operations are modulation, smoothing (i.e., reducing the differences in the data), and sharpening (exaggerating the differences). If the data is discrete, this allows for various additional operations such as sorting and searching.

The introduction of the coding stage allows for a new level of efficiency and speed in processing, transmitting, and interacting with media data and communication content—and this is why first

¹⁷ C. E. Shannon, “A Mathematical Theory of Communication,” *Bell System Technical Journal* 27 (July 1948): 379–423; (October 1948): pp. 623–56.

electronics and later digital computers gradually replaced all other media-specific tools and machines. Operations such as those just mentioned are now used to automatically process the signals and data in various ways—reducing the size of storage or bandwidth needed, improving quality of a signal to get rid of noise, and of course—perhaps most importantly—to send media data over communication networks.

The field of digital image processing began to develop in the second half of the 1950s when scientists and the military realized that digital computers could be used to automatically analyze and improve the quality of aerial and satellite imagery collected for military reconnaissance and space research. (Other early applications included character recognition and wire-photo standards conversion.¹⁸) As a part of its development, the field took the basic filters that were already commonly used in electronics and adapted them to work with digital images. The Photoshop filters that automatically enhance image appearance (for instance, by boosting contrast, or by reducing noise) come directly from that period (late 1950s to early 1960s).

In summary, Photoshop’s seemingly “born digital” (or “software-native”) filters have direct physical predecessors in analog filters. These analog filters were first implemented by the inventors of telephone, radio, television, electronic music instruments, and various other electronic media technologies during the first half of the twentieth century. They were already widely used in the electronics industry and studied in the field of analog signal processing before they were adapted for digital image processing.

Filter > distort > wave

The challenges in deciding in what category to place Photoshop filters persist as we continue going through the Filter menu. The two schemes for classifying software techniques for working with

¹⁸ Even though image processing represents an active area of computer science, and is widely used in contemporary societies, I am not aware of any books or even articles that trace its history. The first book on image processing was Azriel Rosenfeld, *Picture Processing by Computer* (New York: Academic Press, 1969).

media I proposed turn out to be exactly what I suggested them to be—only an initial rough map to start a discussion.

The difficulties in deciding where to place this or that technique are directly related to the history of digital computers as simulation machines. Every element of computational media comes from some place outside of digital computers. This is true not only for a significant portion of media editing techniques—filters, digital paintbrushes and pencils, CAD tools, virtual musical instruments and keyboards, etc.—but also for the most basic computer operations such as sort and search, or basic ways to organize data such as a file or a database. Each of these operations and structures can, both conceptually and historically, be traced to previous physical or mechanical operations and to strategies of data, knowledge and memory management that were already in place before the 1940s. For example, computer “files” and “folders” refer to their paper predecessors already standard in every office. The first commercial digital computers from IBM were marketed as faster equivalents of electro-mechanical calculators, tabulators, sorters and other office equipment for data processing that IBM had already been selling for decades. However, as we already discussed in detail using as examples the work of Ivan Sutherland, and Douglas Engelbart’s and Alan Kay’s labs, whenever some physical operations and structures were simulated in a computer, they were simultaneously enhanced and augmented. This process of transferring physical world properties into a computer while augmenting them continues today—think, for instance of the multi-touch interface popularized by Apple’s iPhone (2007). Thus, while Alan Turing defined the digital computer as a general-purpose simulation machine, when we consider its subsequent development and use, it is more appropriate to think of *a computer as a simulation-augmentation machine*. The difficulty of deciding how to classify different media software techniques is a direct result of this paradigm that underlies the development of what we now call software applications from the very start (i.e. Sutherland’s Sketchpad, 1962–3).

The shift from physical media tools and materials as algorithms designed to simulate the effects of these tools and materials also has another important consequence. As we saw, some Photoshop filters explicitly refer to previous artistic media; others make reference to diverse physical actions, effects, and objects (Twirl, Extrude, Wind, Diffuse Glow, Open Ripple, Glass, Wave, Grain,

Patchwork, Pinch, and others). But in both cases, by changing the values of the controls provided by each filter, we can vary its visual effect significantly on the familiar/unfamiliar dimension. We can use the same filter to achieve a look that may indeed appear to closely simulate the effect of the corresponding physical tool or physical phenomena—or a look which is completely different from both nature and older media and which can be only achieved through algorithmic manipulation of the pixels. What begins as a reference to a physical world outside of the computer if we use default settings can turn into something totally alien with a change in the value of a single parameter. In other words, many algorithms only simulate the effects of physical tools and machines, materials or physical world phenomena when used with particular parameter settings; when these settings are changed, they no longer function as simulations.

For an example, let us analyze the behavior of the Wave filter (located under the Distort submenu). The filter refers to a familiar physical phenomena, and indeed, it can produce visual effects which we would confidently call “waves.” This does not mean that the effect of this filter has to closely resemble the literal meaning of wave defined by a dictionary as “a disturbance on the surface of a liquid body, as the sea or a lake, in the form of a moving ridge or swell.”¹⁹ In our everyday language, we use the word “wave” metaphorically to refer to any kind of periodical movement (“waving a hand”), or any static form that resembles the form of a wave, or a disturbance in the ordinary state of affairs (“making waves.”) According to an influential theory developed by cognitive linguist George Lakoff, such metaphorical use is not an exception, but the norm in human language and thinking. Lakoff proposed that the majority of our abstract concepts are metaphorical projections from sensorimotorial experiences with our own body and the surrounding physical world.²⁰ “Making waves” and other metaphors derived from our perceptual experience of seeing real waves exemplify this general mechanism of language.

If we follow Lakoff’s theory of metaphor, some details of the Wave filter operation—along with many other Photoshop filters

¹⁹ <http://dictionary.reference.com/browse/wave> (August 1, 2012).

²⁰ George Lakoff and Mark Johnson, *Metaphors We Live By* (Chicago: University of Chicago Press, 1980).

Photoshop's Wave filter (version CS5.1). From left to right: the original shape, no filter; the filter with 1 wave generator; the filter with 10 wave generators; the filter with 50 wave generators.

that refer to the physical world—can be understood as similar metaphorical projections. Depending on the choice of parameter values, this filter can either produce effects that closely resemble our perceptual experience of actual physical waves, or new effects that are related to such waves metaphorically.

The filter generates sine wave functions ($y = \sin x$), adds them up and uses the result to distort an image. A user can control the number of sine waves via a parameter called Number of Generators. If this number is set to 1, the filter generates a single sine wave. Applying this single function to an image distorts it using a periodically varying pattern. In other words, the filter indeed generates an effect that looks like a wave.

However, at some point the metaphorical connection to real world waves breaks, and using Lakoff's theory no longer works. If we increase the number of generators (it can go up to 999), the pattern produced by the filter no longer appears to be periodical, and therefore it no longer can be related to real waves even metaphorically.

The reason for this filter behavior lies in its implementation. As we have already explained, when the number of generators is set to 1, the algorithm generates a single sine function. If the option is set to 2, the algorithm generates two functions; if it is set to 3, it generates three functions, and so on. The parameters of each function are selected randomly within the user specified range.

If we keep the number of generators small (2–5), sometimes these random values add up to a result that still resembles a wave; in other cases they do not. But when the number of functions is increased, the result of adding these separate functions with unique random parameters never looks like a wave.

Wave filters can create a practically endless variety of abstract patterns—and most of them are not periodic in an obvious way, i.e. they are no longer visually recognizable as “waves.” What they are is the result of a computer algorithm that uses mathematical formulas and operations (generating and adding sine functions) to create a vast space of visual possibilities. So although the filter is called “Wave,” only a tiny part of its space of possible patterns corresponds to the wave-like visual effects in the real world.

The same considerations apply to many other Photoshop filters that make references to physical media. Similar to the Wave filter, the filters gathered under the Artistic and Texture submenus produce very precise simulations of the visual effects of physical media with a particular range of parameter settings; but when the parameters are outside of this range, these filters generate a variety of abstract patterns.

The operation of these Photoshop filters has important theoretical consequences. Earlier I pointed out that the software tools that simulate physical instruments—paint brushes, pens, rulers, erasers, etc.—require manual control, while the tools that do not refer to any previous media offer higher level automation. A user sets the parameter values and the algorithm automatically creates the desired result.

The same high-level automation underlies “generative” (or “procedural”) software techniques commonly used today. This work generated by algorithms ranges from live visuals and animations by software artist Lia²¹ to the massive procedurally generated world in the videogame Minecraft. Other generative projects use algorithms to automatically create complex shapes, animations, spatial forms, music, architectural plans, etc. (A good selection of interactive generative works and generative animations can be found at <http://www.processing.org/exhibition/>). Since most artworks created with generative algorithms are abstract, artists and theorists like to oppose them to software such as Photoshop and Painter that are widely used by commercial illustrators and photographers in the service of realism and figuration. Additionally, because these applications simulate older manual models of creation, they are also seen as less “new media-specific” than generative software. The

²¹ <http://www.liaworks.com/category/theprojects/>

end result of both of these critiques is that software that simulates “old media” are thought to be conservative, while generative algorithms and artworks are presented as progressive because these are unique to “new media.” (When people claim that artworks that involve writing computer code qualify as “digital art” while artworks created using Photoshop or other media applications do not, they rehearse a version of the same argument.)

However, as they are implemented in media applications such as Photoshop, the software techniques that simulate previous media and the software techniques that are explicitly procedural and use higher-level automation are part of the same continuum. As we saw with the Wave filter, the same algorithm can generate an abstract image or a realistic one. Similarly, particle systems algorithms are used by digital artists and motion graphics designers to generate abstract animations; the same algorithms are also widely used in film production to generate realistic-looking explosions, fireworks, flocks of birds and other physical natural phenomena. In another example, procedural techniques often used in architectural design to create abstract spatial structures are also used in video games to generate realistic 3D environments.

History and actions menus

I started this discussion of Photoshop filters to test the usefulness of two schemes for classifying the seemingly endless variety of techniques available in media software: 1) media-independent *vs.* media-specific techniques (first scheme); 2) the simulations of previous tools *vs.* techniques which do not explicitly simulate prior media (second scheme). The first scheme draws our attention to the fact that all media applications share some genes, so to speak, while also providing some techniques that can only work on particular data types. The second scheme is useful if we want to understand the software techniques in terms of their genealogy and their relation to previous physical, mechanical, and electronic media.

Although the previous discussion highlighted the difficult borderline cases, in other cases the divisions are clear. For example, the Brush Strokes filter family in Photoshop clearly takes inspiration from earlier physical media tools, while Add Noise does not.

The Copy and Paste commands are examples of media-independent techniques; Auto Contrast and Replace Color commands are examples of media-specific techniques.

However, beyond these distinctions suggested by the two schemes I proposed, all software techniques for media creation, editing, and interaction also share some additional common traits that we have not discussed yet. Conceptually, these traits are different from common media-independent techniques such as copy and paste. What are they?

Regardless of whether they refer to some pre-existing instrument, action, or phenomena in the physical world or not, media techniques available in application software are implemented as computer programs and functions. Consequently, they follow the principles of modern software engineering in general. Their interfaces use established conventions employed in all application software—regardless of whether these tools are part of spreadsheet software, inventory management software, financial analysis software, or web design software. The techniques are given extensive numerical controls; their settings can be saved and retrieved later; their use is recorded in a History window so it can be recalled later; they can be used automatically by recording and playing Actions; and so on. (The terms “History palette” and “Actions” refer to Photoshop, but the concepts behind them are found in many other software applications.) In other words, they acquire the full functionality of the modern software environment—functionality that is significantly different from that of physical tools and machines that existed previously. Because of these shared implementation principles, all software applications are like species that belong to the same evolutionary family, with media software occupying a branch of the tree.²²

The pioneers of media software aimed to extend the properties of media technologies and tools they were simulating in a computer—in each case, as formulated by Kay and Goldberg, the goal was to create “a new medium with new properties.” Consequently, software techniques that refer to previous physical, mechanical, or electronic tools and creative processes are also “new media”

²² Contemporary biology no longer uses the idea of an evolutionary tree; the “species” concept has similarly proved to be problematic. Here I am using these terms only metaphorically.

because they behave so differently from their predecessors. We now have an additional reason to support this conclusion. New functionality (for instance, multiple zoom levels), the presence of media-independent techniques (copy, paste, search, etc.) and standard interface conventions (such as numerical controls for every tool, preview option, or commands history) further separate even the most “realistic” media simulation tool from its predecessor.

This means that to use any media authoring and editing software is to use “new media.” Or, to unfold this statement: *all media techniques and tools available in software applications are “new media”—regardless of whether a particular technique or program refers to previous media, physical phenomena, or a common task that existed before it was turned into software.* To write using Microsoft Word is to use new media. To take pictures with a digital camera is to use new media. To apply the Photoshop Clouds filter (Filters > Render > Clouds) that uses a purely automatic algorithmic process to create a cloud-like texture is to use new media. To draw brushstrokes using the Photoshop brush tool is to use new media.

In other words, regardless of where a particular technique would fall in our classification schemes, all these techniques are instances of one type of technology—interactive application software. And, as Kay and Goldberg explained in their 1977 article quoted earlier, interactive software is qualitatively different from all previous media. Over the next thirty years, these differences became only larger. Interactivity; customization; the possibility to both simulate other media and information technologies and to define new ones; processing of vast amounts of information in real-time; control and interaction with other machines such as sensors; support of both distributed asynchronous and real-time collaboration—these and many other functionalities enabled by modern software (of course, working together with middleware, hardware, and networks) separate software from all previous media and information technologies and tools invented by humans.

Layers palette

For our final analysis, we will go outside the Filter menu and examine one of the key features of Photoshop that originally

differentiated it from many “consumer” media editors—the Layers palette. The Layers feature was added to Photoshop 3.0, released in 1994.²³ To quote Photoshop Help, “Layers allow you to work on one element of an image without disturbing the others.”²⁴ From the point of view of media theory, however, the Layers feature is much more than that. It redefines both how images are created and what an “image” actually means. What used to be an indivisible whole becomes a composite of separate parts. This is both a theoretical point, and the reality of professional design and image editing in our software society. Any professional design created in Photoshop is likely to use multiple layers (in Photoshop CS4, a single image can have thousands of layers). Since each layer can always be made invisible, layers can also act as containers for elements that potentially may go into the composition; they can also hold different versions of these elements. A designer can control the transparency of each layer, group them together, change their order, etc.

Layers change how a designer or an illustrator thinks about images. Instead of working on a single design with each change immediately (and in the case of physical media such as paint or ink, irreversibly) affecting this image, s/he now works with a collection of separate elements. S/he can play with these elements, deleting, creating, importing and modifying them, until s/he is satisfied with the final composition—or a set of possible compositions that can be defined using Layer Groups. And since the contents and the settings of all layers are saved in an image file, s/he can always come back to this image to generate new versions or to use its elements in new compositions.

The layers can also have other functions. To again quote Photoshop CS4’s online Help, “Sometimes layers don’t contain any apparent content. For example, an adjustment layer holds color or tonal adjustments that affect the layers below it. Rather than edit image pixels directly, you can edit an adjustment layer and leave the underlying pixels unchanged.”²⁵ In other words, the layers may contain editing operations that can be turned on and off, and re-arranged in any order. An image is thus redefined as a provisional composite of both content elements and various

²³ http://en.wikipedia.org/wiki/Adobe_Photoshop_release_history

²⁴ http://help.adobe.com/en_US/Photoshop/11.0/ (October 9, 2011).

²⁵ *Ibid.*

modification operations that are conceptually separate from these elements.

We can compare this fundamental change in the concept and practice of image creation with a similar change that took place in mapping—a shift from paper maps to GIS. Just as all media professionals use Photoshop, today the majority of professional users who deal with physical spaces—city offices, utility companies, oil companies, marketers, hospital emergency teams, geologists and oceanographers, military and security agencies, police, etc.—use GIS systems. Consumer mapping software such as Google Maps, Microsoft Bing Maps and Google Earth can be thought of as very simplified GIS systems. They do not offer the features that are crucial for professionals such as spatial analysis. (An example of spatial analysis is directing software to automatically determine the best positions for new supermarkets based on existing demographic, travel, and retail data.)

GIS “captures, stores, analyzes, manages, and presents data that is linked to location.”²⁶ The central concept of GIS is a stack of data layers united by common spatial coordinates. There is an obvious conceptual connection to the use of layers in Photoshop and other media software applications—however, GIS systems work with any data that has geospatial coordinates rather than only images positioned on separate layers. The geospatial coordinates align different data sets together. As used by professionals, “maps” constructed with GIS software may contain hundreds or even thousands of layers. The layers representation is also used in consumer applications such as Google Earth. However, while in professional applications such as ArcGIS users can create their own layered maps from any data sources, in Google Earth users can only add their own data to the base representation of Earth that is provided by Google and cannot be modified.

In the GIS paradigm, space functions as a *media platform* which can hold all kinds of data types together—points, 2D outlines, maps, images, video, numerical data, text, links, etc. (Other types of such media platforms commonly used today are databases, web pages, and spaces created via 3D compositing that I will discuss later in the book). In Photoshop the layers are still conceptually

²⁶ <http://en.wikipedia.org/wiki/GIS> (October 9, 2011).

subordinated to the final image—when you are using the application, it continuously renders all visible layers together to show this image. So although you can use a Photoshop image as a kind of media database—a way to collect together different image elements—this is not the intended use (you are supposed to use separate programs such as Adobe Bridge or Aperture to do that). GIS takes the idea of a layered representation further. When professional users work with GIS applications, they may never output a single map that would contain all the data. Instead, users select the data they need to work with at that moment and then perform various operations on this data (practically, this means selecting a subset of all data layers available). If a traditional map offers a fixed representation, GIS, as its name implies, is an information system: a way to manage and work with a large sets of separate data entities linked together—in this case, via a shared coordinate system.

From programming techniques to digital compositing

What is the conceptual origin of Layers in Photoshop? Where do Layers belong in relation to my taxonomies of software-based media techniques? Thinking about various possible sources of this concept and also considering how it relates to other modern media editing techniques takes us in a number of different directions. First of all, layers are not specific to raster image editors such as Photoshop; this technique is also used in vector image editors (Illustrator), motion graphics and compositing software (After Effects), video editors (Final Cut), and sound editors (Pro Tools). In programs that work with time-based data—sound editors, animation and compositing programs, and video and film editors—layers are usually referred to as “channels” or “tracks”; these different terms point to particular physical and electronic media which a corresponding digital application has replaced (analog video switchers, multitrack audio recorders). Despite the difference in terms, the technique functions in the same way in all these applications: a final composition is a result of a “adding up” data (technically, a composite) stored in different layers/channels/tracks.

Photoshop Help explains Layers in the following way: “Photoshop layers are like sheets of stacked acetate. You can see through transparent areas of a layer to the layers below.” (Photoshop CS4 Help, “About layers.”²⁷) Although not explicitly named by this Help article, the reference here is to the standard technique of twentieth-century commercial. Like a film camera mounted above the animation stand, Photoshop software is “shooting” the image created through a juxtaposition of visual elements contained on separate layers.

It is not surprising that Photoshop Layers are closely related to twentieth-century visual media techniques such as cel animation, as well as to various practices of pre-digital compositing such as multiple exposure, background projection, mattes in filmmaking, and video keying.²⁸ However, there is also a strong conceptual link between image Layers and twentieth-century music technology. The use of layers in media software to separate different elements of a visual and/or temporal composition strongly parallels the earlier practice of multitrack audio recording. The inventor of multitrack recording was the guitarist Les Paul; in 1953 he commissioned Ampex to build the first eight-track recorder. In the 1960s multi-track recorders were already being used by Frank Zappa, the Beach Boys, and the Beatles; from that point on, multitrack recording became the standard practice for all music recording and arranging.²⁹ Originally a bulky and very expensive machine, a multi-track recorder was eventually simulated in software and is now available in many applications. For instance, since 2004 Apple has included the multitrack recorder and editor GarageBand on all its new computers. Other popular software implementations include the free application Audacity and the professional-level application Pro Tools.

Finally, yet another lead links Layers to a general principle of modern computer programming. In 1984, two computer scientists Thomas Porter and Thomas Duff working for ILM (Industrial Light and Magic, a special effects unit of Lucasfilm) formally

²⁷ http://help.adobe.com/en_US/Photoshop/11.0/ (October 9, 2011).

²⁸ The chapter “Compositing” in *The Language of New Media* presents an “archeology” of digital compositing that discusses the links between these earlier technologies. Lev Manovich, *The Language of New Media* (The MIT Press, 2001.)

²⁹ See http://en.wikipedia.org/wiki/Multitrack_tape_recorder and http://en.wikipedia.org/wiki/History_of_multitrack_recording

defined the concept of *digital compositing* in a paper presented at SIGGRAPH.³⁰ The concept emerged from the work ILM was doing on special effects scenes for 1982's *Star Trek II: The Wrath of Khan*. The key idea was to render each separate element with a matte channel containing transparency information. This allowed the filmmakers to create each element separately and then later combine them into a photorealistic 3D scene.

Porter and Duff's paper makes an analogy between creating a final scene by compositing 3D elements and assembling separate code modules into a complete computer program. As Porter and Duff explain, the experience of writing software in this way led them to consider using the same strategy for making images and animations. In both cases, the parts can be re-used to make new wholes:

Experience has taught us to break down large bodies of source code into separate modules in order to save compilation time. An error in one routine forces only the recompilation of its module and the relatively quick reloading of the entire program. Similarly, small errors in coloration or design in one object should not force "recompilation" of the entire image.³¹

The same idea of treating an image as a collection of elements that can be changed independently and re-assembled into new images is behind Photoshop Layers. Importantly, Photoshop was developed at the same place where the principles of digital compositing were defined earlier. Brothers Thomas and John Knoll wrote the first version of the program when Thomas took a six-month leave from the PhD program at the University of Michigan in 1988 to join his brother who was then working at ILM.

This link between a popular software technique for image editing and a general principle of modern computer programming is very telling. It is a perfect example of how all elements of the modern media software ecosystem—applications, file formats, interfaces, techniques, tools and algorithms used to create, view, edit, and share media content—have not just one but two parents,

³⁰ Thomas Porter and Tom Duff, "Compositing Digital Images," *Computer Graphics*, vol. 18, no. 3 (July 1984): p. 253–9.

³¹ *Ibid.*

each with their own set of DNA: media and cultural practices on the one hand, and software development on the other.

In short, through the work of many people, from Ivan Sutherland in early 1960s, to the teams at ILM, Macromedia, Adobe, Apple and other companies in the 1980s and 1990s, *media becomes software*—with all the theoretical and practical consequences such a transition entails. This section dives into Photoshop's Filter and Layers menus to discuss some of these consequences—but more still remain to be uncovered.

There is only software

What exactly is “new media” and how is it different from “old media”? Academics, new media artists, and journalists have been writing extensively about this question since the early 1990s. In many of these discussions, a single term came to stand for the whole range of new technologies, new expressive and communicative possibilities, new forms of community and sociality that were emerging around computers and Internet. The term is “digital.” It received its official seal of approval, so to speak, in 1996 when the director of MIT Media Lab Nicholas Negroponte collected his *Wired* columns into the book that he named *Being Digital*.³² Many years later, the term “digital” still dominates both popular and academic understanding of what new media is about.

When I did Google searches for “digital,” “interactive,” and “multimedia” on August 28, 2009, the first search returned 757 million results; the other two only returned 235 and 240 million respectively. Making searches on Google Scholar produced similar results: 10,800,000 for “digital,” 4,150,000 for “web,” 3,920,000 for “software,” 2,760,000 for “interactive,” 1,870,000 for “multimedia.” Clearly, Negroponte was right: we have become digital.

I do not need to convince anybody today about the transformative effects the Internet, the web, and other technological networks have already had on human culture and society. However, what I do want to convince you of is the crucial role of another part of the computer revolution that has been less discussed. And yet, if we

³² Nicholas Negroponte, *Being Digital* (Vintage, 1996).

really want to understand the forms of contemporary media and also what “media” means today, this part is crucial. The part in question is software.

None of the new media authoring and editing techniques we associate with computers are simply a result of media “being digital.” The new ways of media access, distribution, analysis, generation, and manipulation all come from *software*. Which also means that they are the result of the particular choices made by individuals, companies, and consortiums who develop software—media authoring and editing applications, compression codecs, file formats, programming and scripting languages used to create interactive and dynamic media such as PHP and JavaScript. Some of these choices determine conventions and protocols which define modern software environments: for instance, “cut” and “paste” commands built into all software running under the Graphical User Interface and its newer versions (such as iPhone OS), or one-way hyperlinks as implemented in World Wide Web technology. Other choices are specific to particular types of software (for instance, illustration programs) or individual software packages.

If particular software techniques or interface metaphors which appear in one application—be it a desktop program, web application, or mobile app—become popular with its users, they may often soon appear in other apps. For example, after Flickr added tag clouds to its interface, they soon appeared on numerous other websites. The appearance of particular techniques in applications can also be traced to the economics of the software industry—for instance, when one software company buys another company, it may merge its existing package with the software from the company it bought. For instance, in 1995 Silicon Graphics bought two 3D computer graphics suites—Wavefront and Alias—and merged them into a new product called Alias|Wavefront. Big companies such as Google and Facebook are periodically buying smaller companies and then adding the software products these companies develop to their own offerings. Thus, one of Google’s most popular applications, Google Earth is based on software originally developed by Keyhole, Inc. and acquired by Google in 2004.

Often, techniques created for one purpose later migrate into another area, as happened when image processing techniques made their way into Photoshop in the late 1980s. These techniques, developed in the second half of the 1950s for the analysis of

reconnaissance photographs, are now used to creatively modify images and to make photographs more “artistic looking.”

All these software mutations and new species of software techniques are deeply social—they do not simply come from individual minds or from some “essential” properties of a digital computer or a computer network. They come from software developed by groups of people, marketed to large numbers of users, and then constantly refined and expanded to stay competitive in relation to other products in the same market category. (Google and Facebook update their code a few times a day; GitHub, the popular software hosting services, updates its code dozens of times a day.)

In summary: the techniques, the tools, and the conventions of media software applications are not the result of a technological change from “analog” to “digital” media. The shift to digital enables the development of media software—but it does not constrain the directions in which it already evolved and continues to evolve. They are the result of intellectual ideas conceived by the pioneers working in larger labs, the actual products created by software companies and open source communities, the cultural and social processes set up when many people and companies start using it, and software market forces and constraints.

This means that the terms “digital media” and “new media” do not capture very well the uniqueness of the “digital revolution.” (I like the term “media computing”—however it is not used widely apart from some communities in computer science primarily in Europe). Why do they not work? Because all the new qualities of “digital media” are not situated “inside” the media objects. Rather, they all exist “outside”—as commands and techniques of media viewers, authoring software, animation, compositing, and editing software, game engine software, wiki software, and all other software “species.” While digital representation makes it possible for computers to work with images, text, 3D forms, sounds and other media types in principle, it is the software that determines what we can do with them. So while we are indeed “*being digital*,” *the actual forms of this “being” come from software.*

Accepting the centrality of software puts in question another fundamental concept of aesthetic and media theory—that of the “properties of a medium.” What does it mean to refer to a “digital medium” as having “properties”? For example, is it meaningful

to talk about unique properties of digital photographs, electronic texts, or websites? In their article, Kay and Goldberg do pair the words “properties” and “medium” together: “It [electronic text] need not be treated as a simulated paper book since this is a new medium with new properties.” I have also frequently used this combination of words—but it is now time to ask if it is only an alias that can point us to a more precise concept.

Strictly speaking, while it is certainly convenient to talk about properties of websites, digital images, 3D models, GIS representations, etc., it is not accurate. Different types of digital content do not have any properties by themselves. *What as users we experience as properties of media content comes from software used to create, edit, present, and access this content.*

This includes all media authoring and viewing application software made for both professionals and consumers, from Photoshop to your mobile web browser. (It also includes custom software developed for particular products such as a DVD menu or an interactive kiosk.) So while I will continue to use the term “properties” as a shortcut, you should always remember that it stands for *software techniques defined to work on particular types of media ecologies, content and media data*. (Flickr’s whole system for uploading, tagging, organizing, commenting, and sharing images is an example of “media ecology”; a raster 24-bit image stored in JPEG format is an example of a type of “media data.”)

It is important to make clear that I am not saying that today all the differences between media types—continuous tone images, vector images, simple text, formatted text, 3D models, animations, video, maps, music, etc.—are completely determined by application software. Obviously, these media data types have different representational and expressive capabilities; they can produce different emotional effects; they are processed by different sensors and networks of neurons in the brain; and they are likely to correspond to different types of mental processes and mental representations. These differences have been discussed for thousands of years—from ancient philosophy and classical aesthetic theory to modern art and contemporary neuroscience. What I am arguing in this book is something else. Firstly, *interactive software adds a new set of operations which can be applied to all media types—which we as users experience as their new “properties.”* (The examples include the ability to display the same data structure in different

ways, hyperlinking, visualization, searchability, and findability.) Secondly, *the “properties” of a particular media type can vary dramatically depending on the software application used for its authoring and access.*

Let us go through one example in detail. As an example of a media type, we will use a photograph. In the analog era, once a photograph was printed, all the information was “fixed.” Looking at this photograph at home, in an exhibition, or in a book did not affect this information. Certainly, a photographer could produce a different print with a higher or a lower contrast or use a different paper—but this resulted in a physically different object, i.e., a new photographic print that contained different information. (For example, some details were lost if the contrast was increased.)

So what happens with a digital photograph? We can take a photo with a dedicated digital camera or capture it with a mobile phone, or scan it from an old book. In every case, we end up with a file that contains an array of pixels which hold color values, and a file header that specifies image dimensions, color profile, information about the camera and shot conditions such as exposure, and other metadata. In other words, we end up with what is normally called “digital media”—a file containing numbers which represent the details of some scene or an object.

However, unless you are a programmer, you never directly deal with these numbers. Instead, most of us interact with digital media files via some application software. And depending on which software you use, what you can do with a particular digital media file can change dramatically. MMS (multimedia messaging service) software on your phone may simply display a photo sent by a friend—and allow you to forward it to somebody else—but nothing else.

Free media viewers/players that run on desktops or on mobile platforms typically give you more functions. For instance, a desktop version of Google’s Picasa 3.0 includes crop, auto color, red eye reduction, variety of filters (soft focus, glow, etc.) and a number of other functions. It can also display the same photo as color or black and white—without modifying the actual digital media file.

Finally, if I open the same photo in Photoshop, I can instruct Photoshop to automatically replace some colors in a photo with others, make visible its linear structure by running an edge

detection filter, blur it in a dozen of different ways, composite with another photo, and perform hundreds of other operations.

To summarize this discussion, let me make a bold statement. *There is no such thing as “digital media.” There is only software—as applied to media (or “content”).* Or, to put this differently: *for users who only interact with media content through application software, the “properties” of digital media are defined by the particular software as opposed to solely being contained in the actual content (i.e., inside digital files).*

“Digital media” is a result of the gradual development and accumulation of a large number of software techniques, algorithms, data structures, and interface conventions and metaphors. These techniques exist at different levels of generality ranging from a small number of very general (“media-independent”) ones to thousands of very particular ones designed to do particular tasks—for example, algorithms used to generate natural-looking landscapes or software which can extract the camera position from live action footage in order to correctly align a 3D model when it is composited with this footage.

Because of the multiplicity and variety of these software techniques, it is unwise to try to reduce “digital media” to a small set of new properties. Such reduction would only be possible if we could organize all these techniques hierarchically, seeing them as different applications of a few general principles. After thinking on and off about this for ten years (starting with my 1999 article “Avant-Garde as Software” where I first tried to provide a taxonomy of these new techniques) I eventually came to the conclusion that such hierarchy will only mislead us. The reason is that all these techniques equally change the identity of any media type to which they are applied.

The fact that one technique may appear in many software packages designed to work with different media types (what I called “media-independent” techniques) while another technique may be specific to a particular type of media (I called these techniques “media-specific”) does not make the latter any less theoretically important than the former. For instance, because a zoom function is present in word processors, media viewers, animation software, 3D modeling software, web browsers, etc., this does not make it more important than the algorithm designed to do only one particular thing in relation to one media type—for

instance, a “spherize” command which modifies coordinates of all the points in a 3D polygonal model so it appears more spherical.

I do not think that we can qualitatively measure the practical effects on cultural production of both types of operations in this example to conclude that one is more radical than the other. Both operations change the media they act upon qualitatively, rather than quantitatively. They both add new qualities (or “affordances”) to media which it did have before. A Word document which can be zoomed across multiple scales to reveal many pages at once has a different “media identity” from one which cannot. Similarly, the ability to precisely spherize a 3D model is a new way of working with a spatial form that did not exist before 3D software.

In *Avant-Garde as Software* I grouped all new techniques of digital media into four types based on what functions they support: access, generation, manipulation, and analysis. But even such simple differentiation appears problematic to me today—partly because of the evolution of software since 1999, which led to a gradual integration of these functions. For example, when a user selects a media file on his/her laptop, tablet or a phone, the file automatically opens in a media player/viewer program. And today most media viewers and players (Windows Media Player, Apple’s QuickTime Player, etc.) offer some basic editing functions. Therefore, in practical terms today you cannot simply “access” media without automatically being offered some ways to “modify” it. (To be clear, I am talking here about personal computers and mobile devices and not specialized hardware specifically designed to offer only access, and to prevent modification of commercial digital content—such as DVD players or MP3 players.)

How did we arrive at this new situation where instead of looking at or reading content directly, most of us always experience it through the layer of applications? The seemingly obvious answer is the adoption of numerical code as the new universal intermediary. I call it intermediary because in order to make media accessible to our senses, it has to be analog—a travelling wave of oscillating pressure which we experience as sound, the voltage levels applied to the pixel elements of an LCD which makes them appear as different colors, different amounts of dyes deposited on paper by dye-sublimation printers, and so on. Such conversions from A to D (analog to digital) and D to A (digital to analog) are central for digital media functioning: for example, from light

waves to numbers stored in a file representing the image, and then back to the voltage levels controlling the display. Or, in another example, when we design an object to be printed on a 3D printer, an analog representation on the screen is translated by a computer into a digital file that then drives the analog signals controlling the printer.

The two levels of encoding—first, a sampling of a continuous analog signal to create its representation using a scale of discrete numbers (for example, 256 levels commonly used to represent grey tones in images), followed by a translation of this discrete representation into a binary numerical system—make “media” incomprehensible for direct observation. The main reason for this is not the binary code *per se* (invented by the Indian scholar Pingala between the fifth and second century BC) since it is possible to learn how to convert in your head a binary notation into a decimal one. The problem is that representing even one image digitally requires lots of numbers. For example, an image in HD resolution (1920 × 1080) contains 2,073,600 pixels, or 6,220,800 distinct RGB values—making it very hard to comprehend the patterns such a set of numbers may represent if you examine them directly. (In passing: because of these considerations, any digital image can be understood as information visualization—revealing patterns contained in its numerical representation.)

Because looking at such sets of numbers with our bare eyes is meaningless, we need to employ some technologies to translate them into analog representations acceptable to our senses. Most often, an image file is translated by digital hardware and software into an image appearing on our screen. However, a digital representation of one type of media can also be translated into another media type that is meaningful to our senses. For example, in audio-visual performances software often uses video to drive sound, or reversely uses sound to generate abstract visuals. (Interestingly, the precursor to Edison’s 1877 phonograph—the first device to record and reproduce sound—was Édouard-Léon Scott de Martinville’s 1857 phonautograph that transcribed sound into a visual media. In other words, sound visualization was invented before sound recording and reproduction.)

From the beginning, technologies that generated and transmitted electro-magnetic analog signals (e.g, the gramophone) included at least some controls for its modification such as changing signal

amplitude. The first well-known electronic instrument invented by Léon Theremin in 1920 turned such controls into a new paradigm for music performance. A performer controlled amplitude (volume) and frequency (pitch) of a sound by moving his/her hands closer or further away from the two antennas.

Software significantly extends this principle by including more controls and more ways of representing the data. For example, I can choose to display this text I am writing now in Word as an outline, or select a “Print Layout” which will show me boundaries of pages; I can choose to see footnotes or hide them; I can ask the application to automatically summarize the text; I can change different font families and sizes, and so on. Thus, while the actual data as it is represented and stored in a computer is no longer directly accessible to our senses, the new model of encoding and access has other significant advantages since the data can be formatted in a variety of ways. This formatting can be changed interactively; it can be also stored with the data and recalled later.

This discussion can help us to understand the relations between earlier electro-magnetic recording and reproduction technologies, which were developed in the last decades of the nineteenth century, and media software developed 100 years later. (Telephone: Bell, 1875; phonograph: Edison, 1878; television: Nipkow, 1884; radio: Fessenden, 1900.) While previous reproduction technologies such as woodblock printing, moveable type printing, printmaking, lithography, and photography retained the original form of media, the media technologies of the late nineteenth century abandoned it in favor of an electrical signal. In other words, they introduced *coding* as a way to store and transmit media. Simultaneously, these technologies also introduced a fundamentally new layer of media—*interface*, i.e. the ways to represent (“format”) and control the signal. And this in its turn changes how media functions—its “properties” were no longer solely contained in the data but were now also depend on the interface provided by technology manufacturers.

The shift to digital data and media software 100 years later generalized this principle to all media. With all data types now encoded as sets of numbers, they can only be accessed by users via software applications which translate these numbers into sensory representations. The consequence of this is what we have already discussed: all “properties of digital media” are now defined by the

particular software as opposed to solely being contained in the actual content, i.e. digital files. So what was already true for audio recording, radio, television, and video now also applies to text, images, and 3D.

In short: media becomes software.

I could conclude this chapter here—however we need to do one more thing. It would be unfair to direct all this attention to the term “digital media” without also taking up another term related to it. Today this term is both widely used and also often put into question. The term is “new media.”

Since we now understand that “media” today is really a set of software techniques constantly in development, this gives a new meaning to this troubled term. Just as there is no logical limit to the number of algorithms which can be invented, people can always develop new software techniques for working with media. So from this perspective, the term “new media” captures well this fundamental logic of “the computer metamedium.” *Software-based media will always be “new” as long as new techniques continue to be invented and added to those that already exist.* And while not every one of these techniques will change significantly how particular media or a combination of media will function, many will.

This logic of “permanent extendability” of media software follows the logic of software engineering as a whole. Computer scientists working in the academy and software companies constantly develop modifications of the existing algorithms, apply algorithms from one area in another area, and develop new algorithms. This logic—which can be also called “permanent innovation”—is different from the logic which governed the development of media technologies of the industrial age. Take cinema, for instance, from 1890 to 1990, (i.e. until the adoption of software tools by the film industry). Although the construction of film lenses, the properties and types of film stock, the operations of film camera, and other elements of film technology changed dramatically over the course of the twentieth century, the basic “representational capacity” of this image type remained the same. From Edison to George Lucas, film images continued to be produced in the same way: by focusing light reflected from the objects via lens on a flat plane. This type of capture process creates particular types of images of

visible reality. Objects are rendered in one-point linear perspective. This means that the geometric properties of any scene regardless of its content are subjected to the same type of transformation—the same mapping that preserves some properties of the visible (perspective distortion related to distance of objects from the observer) as opposed to other properties (the real proportions of object sizes, for example). Additionally once a recording was made, the geometry of the image could not be modified later.

Once software enters the picture, these constants of the film medium become variables. While looking no less “real,” film image can now have multiple relationships to the world being imaged. Digital compositing allows for the seamless insertion of 3D computer-generated models that were not present in the original scene. Conversely, the objects that were present can be seamlessly removed from images. Interactive virtual panoramas which allow the user to move around the space can be constructed automatically. In some cases, it is even possible to re-render a film sequence as though it was shot from a different point of view. And these are just some of the new ways in which new software changes film identity. (All these new possibilities, of course, also apply to video.)