Conclusion

How should cultural theory and media theory engage with the ideas and practices of the Creative Coding movement? How should the humanities, the arts, and cultural studies engage with the practice of Creative Coding? Creative Coding follows New Media Art but is more explicitly immersed with informatics and with the effects of informatic technologies on society, the economy, and our lives. Media and cultural theory are already built into Creative Coding. Creative Coding is more than theory – it is a hybrid of theory and practice. This double-sided commitment is what theory wants. As Karl Marx wrote in his eleventh thesis on Feuerbach: "The philosophers have only interpreted the world; the point is to change it."[798]

Students in cultural studies, humanities, design, and the arts should learn to write software code. There should not be a strict border between code and poetry, between inscription that means something to the processor and that which means something to humans. The readability of the source code is very important. The task is not only about learning how to write code in the sense of being able to do what software engineers do. It is about changing what code is. Hyperreality is implemented in detail with code and can be changed with transformed code. Code should be transfigured with philosophical, political, aesthetic, and design knowledge.

In the hybrid pedagogical concept, knowledge or theory or ideas is introduced both in a systematic way as a "long discourse," and in a new way as small "nuggets" of knowledge/theory, brought into relation with a specific design project, artwork, film, computer game, or other cultural artefact (the design of which is connected to that knowledge field). One is continuously on the border between theory and practice. It is a pedagogy of the hybridity of ideas and practice for art and design students. This could be a way of teaching the making of websites (*HTML*, *CSS*, *JavaScript*), interactive art installations (*Processing*), neural network Deep Learning image recognition or chatbots (*TensorFlow*), and virtual world games for VR glasses and the metaverse (*Unity*). Students could develop into software innovators.

How can Creative Coding change computer science itself – in the latter's core concepts, applications, educational curriculum, and in the definition and profile of who is a programmer? Programmers should get a serious education in philosophy, literature, media theory, and art. I have argued that informatics or computer science has both a

scientific and a cultural component. The history of programming languages shows that this is the case since programming progresses from paradigm to paradigm via paradigm shifts. I do not subscribe to the extreme argument often made in Science and Technology Studies that all computer science is cultural.

Turing's and Neumann's original World War II-era computer science is not the same as 1960s COBOL business-procedural programming nor the same as 1980s object-orientation and the Xerox PARC (where Steve Jobs and Bill Gates pilfered their ideas) graphical user interface revolution, and then Artificial Life, quantum computing, biological computing, neural networks, Deep Learning, etc. These are all different paradigms of computing. Add to these the new paradigm of Creative Coding discussed in Part Three of this study. My claim is that the two-level configuration of scientific and cultural components is already evident within the work of Turing and von Neumann. Understanding the history of computer science and its changing paradigms as divided in this way is a prerequisite to freeing the future of informatics or digital technology or Creative Coding as an existentially open-ended undertaking of art, culture, ethics, and expressivity, where we can both respect science and formulate projects for a better future.

Towards a Transdisciplinary Informatics

We need a transdisciplinary informatics that is up to the task of engaging with the situation that we have become an "informatic society." At the time of the mid-twentieth century invention of computer science, no one knew that informatics would have such a major impact on all culture and everyday life. Hence it was normal that computer science back then was a purely technical discipline. But this is no longer the case.

How can we take steps towards a different informatics, towards more "compassionate" and "sensible" software systems and environments? How can this change contribute towards becoming a more ethical, livable, and ecological society? What will the practice of software development be like when its concern is both software codes and cultural codes?

Is a partnership between humans and AI possible? How can AI and posthumanism together be transdisciplinary projects for transforming humanity to become more human? The goal of AI should not be to build so-called autonomous systems which are managed by humans only from the outside. Rather than a dualism between algorithms and morality, there should be an embedding of ethics into the heart of software code. How can computer science become flexible enough to be a conscious and creative cultural practice as well as science and technology? There should be a going beyond the dualism of formal language and expressivity in code. Software code must become poetic, ambivalent, and musically resonant. It must go beyond the so-called discrete logic of conventional programming languages.

The cultural theory of simulation and hyperreality has a lot to learn from the technical-cultural patterns of programming languages. For example, the concept of inheritance hierarchies in object-oriented software design explains a lot about how transmedia visual culture works. There is also creativity in the "live-coding scene" – writing and visually displaying source code in an improvised way during an art installation, performance, or

a group coding collaboration, often in connection with dance, poetry, music, or audio-visual exhibition.

Connecting software codes (which are also systems of notation) to the history and future of writing is an essential project. To look at software poetically is to diverge from the received view that software code is a formal, logical, numeric, combinatorial, and calculating notational system. It is to grasp instead the cultural, linguistic, poetic, aesthetic, resonant, musical, and semiotic aspects of software. To look at software poetically is both to see the history of software retrospectively in this light, and to consciously emphasize the cultural and human-language dimensions of software in future variants of informatics.

Following the German media theory or "media science" or "media archaeology" that was founded by Friedrich Kittler, I employ the term media technology as a synonym for the term computer.[799] I do not wish to choose between focus on the non-historical characteristics of a scientific media technology and the discourse- and narrative-oriented analysis which emanates from the humanities and cultural studies. I seek rather to synthesize the two perspectives. If we place too much emphasis on the scientific, then we risk becoming a conservative force blocking conceptual changes at the deepest level in paradigm shifts. If we place too much emphasis on the cultural, then we risk a relativism that misses the scientific axioms. I seek a balance between – or two-tiered understanding of – the scientific and cultural layers of any science or, specifically, computer science.

A new scientific (or cultural) paradigm replaces (or renders invalid) the previous paradigm. It is rather a supplement. The newer paradigm is also made possible by a conscious internalizing of the previous paradigm, which is not rendered epistemologically *dépassé*, implying instead an *Aufhebung* in the Hegelian sense. Programming has proceeded in time through successive paradigms and paradigm shifts. These technical paradigms are also cultural or knowledge paradigms which parallel distinct stages of a cultural-historical genealogy. The tradition of writing genealogies of such stages was begun by Nietzsche and Foucault.

Thomas S. Kuhn on Paradigm Shifts in Science

In *The Structure of Scientific Revolutions*, Thomas S. Kuhn asserts that, in the history of science, discoveries (novelties of fact) and inventions (novelties of theory) are not so distinct from each other.[800] Important scientific discoveries that incite paradigm shifts belong generally to an era of history and cannot reasonably be attributed only to a specific individual scientist or a single date in time. A new paradigm in any given science does not render the previous paradigm invalid. Copernican astronomy appears to have superseded the astronomical system of Ptolemy. Yet the calculations and predictions of the ancient Greek-Egyptian mathematician were robust and are still widely used today in engineering contexts. The heliocentric discoveries of Copernicus in the sixteenth century and Galileo in the seventeenth century ignited a delayed-reaction paradigm shift. The Copernican model of the sun-earth relationship, which disputed and eventually supplanted the geocentric universe of Ptolemy, was not accepted for centuries due to the anxiety about the loss of our anthropocentric status in the cosmos which it provoked. Humans, cre-

ated in God's image, were no longer the center of the universe. The sun does not revolve around the earth as was previously believed; the earth revolves around the sun. Physical reality and its laws were elevated to a sovereign status in relation to humans.

Kuhn describes how the beginning murmurs of a paradigm shift start to become audible. Anomalies or counter-instances to the prevailing theory occur in the crisis phase. The decision to reject the prevalent paradigm is simultaneous with the decision to embrace the new one. After an interlude of resistance, encompassing various attempts to resolve the crisis quickly through modifications to the existing theory, a change in framework or *Gestalt* perception finds wide acceptance as the way to make sense of the new data. The crisis of an established scientific paradigm can end (in one possible scenario) through the normal science of that paradigm reasserting itself and maintaining its hold on the scientific community at hand; or the crisis comes to be seen as unsolvable (a second possible scenario) and no further resolution is sought in the short term; or finally (in a third scenario), a new candidate for paradigmatic dominance emerges and a battle for hegemony ensues. During the transition period there is an overlap between the approaches to problems of the old and new paradigms.

In his 1969 Postscript, Kuhn states that he intended two different meanings for the term "paradigm."[801] The first meaning is the constellation of group commitments (ideas, tools, and research methods), values, beliefs, and techniques shared by the members of a given scientific community. The second meaning refers to only one element of that constellation: the concrete solutions to puzzles that are encountered in practice, and which end up being shared models or examples of how to apply the consensus theories according to an agreed upon set of rules.

Herbert A. Simon, *The Sciences of the Artificial*

Is computer science a science? What is at stake in the question of "the sciences of the artificial"? Herbert A. Simon was a distinguished professor for five decades at Carnegie Mellon University, one of America's most elite and important technology institutes of higher education. Simon won the Nobel Prize in economics and the Turing Award, which is the most prestigious citation for achievement in computer science, given annually by the Association for Computing Machinery. Simon's work was highly interdisciplinary, ranging from economics and psychology to Artificial Intelligence and the study of large organizations and complex systems. In his book *The Sciences of the Artificial* (a pioneering work first published in 1969, then subsequently revised in 1981 and 1996), Simon does not seek to formulate any fundamental philosophical definition of the array of sciences which study – to invoke his terms – the "man-made" as opposed to what is "given by nature."[802] He searches for ways to pragmatically identify the characteristics of the artificial sciences. His approach is imbued by American pragmatism.

What makes some phenomena and systems artificial, according to Simon, is not only or primarily the fact that they come to be through artifice, design, engineering, or other human cultural practices, but rather the pragmatic and operational circumstance that they interact with their environment. This quality of the objects studied by the sciences of the artificial that they continually engage with their environment renders them more

dynamic and changeable over time than the phenomena studied by the natural sciences. It is more difficult to make statements that remain valid for a long duration of time about synthetic entities.

The contingency and malleability of artificial phenomena, according to Simon, are due to their deep and continuous involvement with their environment, in contrast to the "necessity" which is a chief property of natural phenomena. The latter exist in a subordinate relationship to the power of natural laws. This difference has unfortunately often led scholars and thinkers to mistakenly regard artificial materials and systems as not "falling properly within the compass of science." The challenge that Simon defines is to pinpoint exactly how can one make valid empirical propositions about things and systems which behave varyingly in their permanently changing circumstances and whose behavior is different if observed at different times.

According to Simon, the thorniness of the problem of artificiality – which affects many disciplines extending from economics, management, and information processing to education, engineering, and the cognitive psychology understanding of thinking and problem solving – is also due to the normative character of the objects inquired into by these fields. These sciences are concerned not only with "how things are but with how they might be." There are ethical, political, economic, and purposive-rational goals involved in the investigated occurrences. Thus, Simon elevates design to a central position in his framework. The mission of creating a science of artificial is inseparable from the task of creating a science of design. Design is the key to grasping and intervening into how systems abide in complex environments.

In her 2009 book *Simulation and Its Discontents*, MIT professor of the social study of science and technology Sherry Turkle reflects on the transformations in scientific, engineering, and design education at MIT that occurred when computers and software were introduced to all fields of study in the 1980s and 1990s as major and intensive components of the learning curriculum.[803] Turkle concludes with regret that many thinking skills and significant knowledge were lost in the training of scientists and professionals when all disciplines came to increasingly resemble each other in their shared emphases on simulation and visualization. The way of working – without computers – of older professors who were retiring was more direct and less mediated. Herbert A. Simon, contrary to Turkle, sees the computer as being a fantastic development for its stimulation of interdisciplinarity. He praises "the growing communication among intellectual disciplines that takes places around the computer… All who use computers in complex ways are using computers to design or to participate in the process of design."[804] The computer becomes the tool *par excellence* for transdisciplinary design. In Simon's vision of the inter- or transdisciplinary, there is no place for the abiding value of the mono-disciplines.

What is important about artificial systems for Simon is their goals, functionality, self-organization, normativity, capability to adapt to new circumstances, and orientation towards how things should be. The artefact is performative in its interaction with its environment. Simon sums up his position with the concept of interface. There is an interface or meeting point between the inner organization and the outer environment of the artificial entity which underlies its design or intended purpose. Simon claims that his conceptual framework has the benefit of being predictive. Insights into the goals and behavior of the artifice enable an anticipatory advantage in foreseeing the future. He cites

the state of homeostasis in biology (the steady internal conditions maintained by living systems) as an example to support his thesis.

Simon recommends positing an invariant relationship between inside and outside for heuristic purposes, isolating the inner system from its outer environment in a temporary bracketed way. The interface between inside and outside should be designed simply and elegantly, therefore strengthening its qualities of powerful abstraction and general applicability. The priority of interface then leads to the potency of simulation. Once we have a clear grasp of the interface, then that interface can be simulated "as a technique for achieving understanding and predicting the behavior of systems."[805] The computer, with its ability to simulate, model, and try things over and over, becomes the ideal tool for the sciences of the artificial.

Simulation, according to Simon, can provide amazing amounts of new knowledge. Software can imitate human behavior in many domains, given an algorithmic description of the behavior. "No artifact devised by man is so convenient for this kind of functional description as a digital computer."[806] It is the ideal device for the empirical social sciences, the exploring of the consequences of alternative independent variable values and organizational assumptions. The design of software, according to Simon, is a behavioral process. The software program is a logical arrangement of symbols to be manipulated by a program-control component (a Turing machine). In any design or conceptual phase of the software development cycle, not much can be known about how the software is going to behave. You build the software and then you see later how it behaves. Things become known in doing and trying out, in observing what happens when the software is up and running and interacting with its environment.

Herbert A. Simon provides an empirical methodology for the unification of the social and informational sciences. Simon wants predictability of the entire artificial world-ambience. His argument is a visceral and inaugural rejection of any philosophical approach. He is interested in the acquisition of useful knowledge for the scientific management of that simulation model that we call society. Yet his highly influential approach dissuades us from asking the crucial question: how can the philosophy of science be applied to computer science?

Simon presciently asked the crucial question "What is at stake in the question of 'the sciences of the artificial' as separate from scientific approaches to the natural world"? However, his strictly empirical methodology led him away from all philosophy, and from any engagement with the philosophy of science to help in answering the question.

Two Meanings of Artificial Intelligence

AI has at present two distinct meanings. Recently it has become a matter of business and "data science": Deep Learning, pattern recognition, neural networks, and "Big Data." Originally AI was the idea of a machine capable of thinking. This raised provocative questions for SF, philosophy, and sociology: what would this techno-scientific breakthrough do to society and our lives? Would AI be a danger to humanity? Would a paradigm shift in informatics be required to accomplish AI? The two different meanings of AI are intimately related and inseparable. Ignoring the SF and philosophical questions leads to

AI projects which have the goal of building autonomous systems. These systems are albeit architected and operate within the newer *pattern-based* paradigm of neural network AI and are more adaptive and responsive to their environment than the sequences of programmed instructions of classical *rule-based* informatics. Nonetheless such systems are designed to substitute for and act independently from humans according to the long tradition of automation in twentieth-century capitalism and industry from which they inherit their goals. They are engineering-technology ventures.

Some scientists and engineers who work in AI in the first sense of the technologies of today say that the AI depicted in science fiction films is fantastic and will never happen. This is to miss the point that in those films profound questions are being asked. It seems like an excuse for the scientist or engineer to not consider the philosophical, political, and lifeworld dimensions and implications of the work that they are doing. We need to move beyond the dichotomy between the humanities and the computer sciences. There has been a certain tendency in the humanities to ironically approve of the computer sciences remaining technical and engineering-oriented to keep them as their foil in an oppositional role. This way, the humanities preserve their possession and authority of creativity and consciousness in their asserted contrast to the computer sciences. The reason why renowned humanist philosophers like John Searle (of the famous *Chinese Room Argument*) resisted Artificial Intelligence for so long and said that it is impossible is because of their allegiance to the humanist culture which says that only humans enjoy an enumerated list of certain special and ineffable qualities: consciousness, feelings, experience, emotions, ethics, rational judgment, free will, etc… and robots and androids could never have those qualities.[807] This is anthropocentrism and the establishing of the human-non-human hierarchy of moral worth. We keep technology in its place as machine-like and void of ethics to keep our higher position in the hierarchy.

Posthuman transdisciplinary informatics does not reject the logic of computing, but rather seeks to build on top of that logic, extending computing to be more ambivalent, emotional, embodied, aesthetic, creative, etc. I would like to transcend the dualism between rational/combinatorial/algorithmic intelligence and those special qualities which humanist culture has granted to humans which make them "not technology." To insist that machines are dead inert objects, or that everything about computers and code and software is engineering, is paradoxically to cling to humanism. It is a refusal to move on to the posthuman or cyborg paradigm where humans are in dialog with technology as environment, and humans come to terms with their own existential condition as technology.

Four Key Mistakes of the Artificial Intelligence Mania

There is currently (year 2024) a mania surrounding what is called Artificial Intelligence. There is astonishment about what ChatGPT and similar large language model-based chatbots can do. The range and depth of conversational applications are amazing. Trained-on-Big-Data algorithmic processes and systems based on implementations of the Machine Learning/Deep Learning neural network pattern-based computer science technique are seemingly everywhere. AI text generators and AI image generators have

become so sophisticated that they appear to rival or even threaten human creativity. There is fear about AI becoming more "intelligent" than us, treating us as inferior, and diminishing the aura of what it means to be human.

I will mention what I believe to be four key mistakes of the current AI mania. First, today's AI is falsely seen as being mainly a break from, rather than a continuity with, what informatics has been for the last several decades. AI is ostensibly defined by its difference from other branches of Information Technology (IT) in that its stated goal is the development of machines and assemblages which can think, learn, and interact similarly to humans. Yet the essential question of the "philosophy of technology" has for a very long time been "what is the impact of informatics on society and the lives of citizens of late capitalism?" For the most part, since the first wave of digitalization, with its milestone inventions of the Personal Computer, the Internet, and the smartphone, very little attention has been paid by the public and the "pundits" to the possible deleterious effects of computing on everyday life. Where was all the worrying during the past forty years? AI has practical functions in areas of logistics, economic organization, and management – finance, healthcare, transport – not very different from the previous generation of IT applications.

The second key mistake is to regard Artificial Intelligence as a development in the abstract without seeing the context of its embeddedness in capitalism. We should not be talking in an alarmist apocalyptic way about a potential dreaded "AI takeover" in the future because AI already runs the world as an instrument and coding of the power that the big corporations wield over our lives. We are ruled by the access to our personal data that platform and surveillance capitalism have, as well as by their control over what we browse and see in the so-called "attention economy." We are increasingly addicted to our electronic devices. We are ideationally isolated and polarized in our discursive filter bubbles and echo chambers, immersed in "fake news" and conspiracy theories, and driven by our anonymous intense emotional hatred of others. The racial and income-level discrimination or bias present in the data training sets of the informatic-capitalist economy find their way into the AI algorithms. The code of AI is not mimetic of some ahistorical generic human intelligence but is rather derived from the historically hierarchical and asymmetrical power relationship between capital and labor that drove previous rounds of the automation of work and knowledge, such as Ford's assembly line and Frederick W. Taylor's "scientific management."

The third key mistake is to assume that Artificial Intelligence primarily means the development of so-called autonomous systems which operate independently from human decision-making. What I advocate instead throughout the current book is that we should rather think about the design and implementation of AI systems ethically as the sharing of responsibility between humans and non-human technological actors in society, the economy, and political governance. Autonomous Artificial Intelligence (AAI) is promoted with enthusiasm especially by those with an engineering or money-making business mindset. This technology is seen as making it possible for intelligent machines to carry out complex tasks with no human intervention, thus streamlining efficiency and increasing profits. The conversation about autonomy versus collaboration is reduced to a purely technical discourse. Here we have a blatant example of ignoring moral, social, political, and ecological considerations, as well as the warnings emanating from the cultural

imaginary of AI as emblematized in SF narratives. What we should seek is partnership between humans and AI.

The fourth and final key mistake is that the fear of Artificial Intelligence becoming power-hungry, as expressed in many SF films, and in the discourses surrounding Superintelligence, the Singularity, and the dreaded "AI takeover" are psychological projections of the all-too-human characteristic of power-hungriness and the human history of violence. As Captain James T. Kirk says, echoing the existentialist philosophy of radical freedom, in the coda of the *Star Trek: The Original Series* episode "A Taste of Armageddon":

> We're human beings with the blood of a million savage years on our hands, but we can stop it. We can admit that we're killers, but we're not going to kill today. That's all it takes, knowing that we're not going to kill today.

Agencies of liberal political states or trans-states like the European Union are having a very difficult time figuring out how to "regulate AI." The technology (exemplified by the sensational impact of ChatGPT) advances more rapidly than the politicians' or experts' understanding of it. The EU wants to control the potential harms of AI (bias and discrimination built into algorithms, the spread of "post-truth" disinformation, the elimination of jobs, etc.) without making the mistake of interfering with its economic benefits. What these policymakers do not "get" is that AI is a paradigm shift in informatics from rule-based to pattern-based logic and actions. The big corporations (capitalism) have already grasped and carried out this shift. To keep up, liberal political philosophy needs to make the same shift. Rules and regulations are no longer the way to go. The way to counter the negatives of AI is with a morally good counter-AI. Get patterns out there to counter the patterns of capitalism.

Andreas Reckwitz's Objection to "Creativity"

In his book *Anti-Media: Ephemera on Speculative Arts*, Florian Cramer says that "most artists and designers despise the word *creative*."[808] According to Cramer, those who use this term are either artists who make pretty things like decorative pottery or high-income earners whose expressivity is hopelessly co-opted by the so-called "creative industries" of brutal yet "progressive" neoliberal capitalism. These creative industries have absorbed and commodified gestures of rebellion, experimentation, hipness, and non-conformity into advertising and the imagery surrounding "cool" daily life work practices in high-tech white-collar jobs.[809]

According to prominent German sociologist of culture Andreas Reckwitz in his book *Die Gesellschaft der Singularitäten: Zum Strukturwandel der Moderne*, a so-called "creativity dispositive" has already replaced work, production, and profit as the main engine driving late capitalism.[810] For Reckwitz, creativity is an economic-cultural *invention*. It is something that is now expected of those successfully integrated as the affluent stratum of the post-industrial capitalist system. There is a social regime of "the aesthetic new." Given the widespread influence of neo-Marxist sociological theses such as those of Reckwitz, I am aware that some readers of the present study will be skeptical of my usage of the

term *Creative Coding*. In my view, the empirical critique of the recuperation of creativity by capitalism is justified and important. Yet this analysis should not lead to a universal rejection of all creativity. On the contrary, the critique should be a step towards reflecting on how creativity can be reinvented in the context of a consciously *anti-capitalist* or *post-capitalist* intention.

Reckwitz correctly points out that so-called "creativity" has become a driving *avant-garde* force in the "creative industries" of neo-liberal capitalism. However, to elevate that into an argument against all creativity would be bad faith and is not logically valid.

Jaron Lanier's Phenotropic Programming

In his autobiographical work *Dawn of the New Everything: A Journey Through Virtual Reality*, VR pioneer and founder of the company Visual Programming Languages Jaron Lanier explains his view of software code which has a lot of overlap with the view laid out in the present study.[811] Lanier references Admiral Grace Hopper – the inventor of the first *linker* (program to convert human-readable code to machine-readable code) and one of the great pioneers of the early history of computer programming. He explains that Hopper originated many of the "core patterns for how software is still created today," such as the duality between source code and the executable, the back-and-forth alteration between writing code and testing the running program, the *compiler*, and the hierarchy of assembler and high-level languages.[812] Such artefacts and practices, asserts Lanier, are fundamentally arbitrary. They are the result of specific design decisions which could historically have gone another way. The work patterns and steps for developing software could easily be completely different.

The decisions made during the history of computing regarding how programming would be done were made on the cultural level and not on the scientific level. Lanier writes:

> There was never a reason to think… all software always had to follow the pattern set by Hopper… The only things that are fundamental and inviolable – truly real – while you are using a computer are you and the run of patterns of bits inside the computers. The abstractions linking those two real phenomena are not real.[813]

Everything between you and the hardware is a cultural decision. Lanier has a justified complaint about software programming being too obsessively exact: "You have to become a robot to program a robot."[814] He imagines a completely different practice of programming which might have come about. This was the vision of his 1980s company. He would like a scenario "where you could paint and repaint the bits on a screen, so that a program could be redone as it was running."[815] You could change all the rules in real-time and on the fly while inside the real-slash-simulation software. You would be immersed in Virtual Reality and melded in partnership with the virtual world or game – rather than being the programmer-subject locked in a dualistic anthropocentric controlling relationship with the program. Programming would be more artistic, intuitive, symbolic, and experimental. Lanier writes:

> I suspect that if computer programming had evolved along these lines, the whole society would be different today… A more concrete, visual, and immediately editable style of computation would be modeless and better suited to VR. You would be able to change the world while you are inside.[816]

Lanier calls this new way of programming "phenotropic" – which means surfaces turning towards each other. Two entities interact with each other's surfaces via pattern recognition observation. He cites music as his inspiration for user interface design and software expressivity. "The programming of the future will have to be a lot like jazz."[817]

Creative Coding and Radical Software

Creative Coding refers to software tools, programming languages, and hardware platforms which are intended for and developed by artists and other creatives. An example is the "Integrated Development Environment" (IDE) called *Processing*, which was originated by Casey Reas and Ben Fry.[818] *Processing* helps to make interactive visual art projects. In "generative art," artworks are created using an autonomous system such as a computer, a robot, or an algorithm. Other examples of Creative Coding include music programming languages like *SuperCollider*, and microprocessors for learning like *Arduino* and *Raspberry Pi*, which can control electronic devices and help make new media art installations.[819] A line of code is an aesthetic artefact and not only an instruction to the machine. New software layers open performance spaces for music, poetry, storytelling, and dance.

Radical Software is a diverting of technologies in the sense of the Situationist practice of *le détournement* (the "detouring" of something from its original use).[820] One creatively overturns and transfigures the intended designs and uses of digital media technologies in the mainstream. Online existence and Augmented Realty are ambivalent interspaces or contested arenas poised between hyperreality and transformative potential, inscribed via software code.

Radical Software is poetic, expressive, ambivalent, and resonant. It emphasizes the writerly qualities of the code beyond the code as the means to a functional end. Radical Software operates in the double territory-and-imagination of material-and-informational space. As Walter Benjamin already wrote in 1935 in "The Work of Art in the Age of Its Technological Reproducibility," cinematic special effects alter the dimensionality of what we experience as space.[821] "With the closeup," Benjamin writes, "space expands; with slow motion, movement is extended."[822] Once media technology has passed beyond a certain threshold, then space is no longer strictly a physical-geographical-architectural space. Space must be rethought as a dynamic hybrid of what was previously called "real" and what was previously called "virtual." Space is both real-physical and simulated-virtual.

What Does "Software" Mean?

The journal *Radical Software* was started in New York City in 1970 by a group of artists, writers, and filmmakers who gravitated around the Raindance Corporation, an "alternative media think tank" that had been founded the previous year by Frank Gillette.[823] The magazine (eleven issues were published altogether) incited the growth of a community of video artists. Its theoretical articles focused on critique of the centralized corporate power which controlled the television industry and the dominant mass media structures in America. The publication also offered practical information about making videos with low-cost camera equipment such as the Sony Portapak (the first portable video recording system, introduced in 1967) and experimental video aesthetics. There were discussions of the ideas of thinkers like Gregory Bateson, Buckminster Fuller, and Marshall McLuhan. Some of the topics addressed were ecological issues, proposals for decentralizing media and increasing access to information, and the philosophy of technology. The journal inspired the publication of a landmark video art book (*Video Art: An Anthology*, edited by Ira Schneider and Beryl Korot) and a book about video political activism (*Guerrilla Television* by Michael Shamberg).[824]

As Davidson Gigliotti writes at the website where all published issues of *Radical Software* have recently become available online, a study of the history of the journal shows that video art – arguably the first sub-genre of what would later become New Media Art – had its origins in a critical and utopian view of the present and possible future of media in American society. The *Radical Software* collective had a vision of an alternative to the commercial television industry. They wanted TV to become a creative and democratic media.

As their choice of the word *software* for the name of the journal shows, the *Radical Software* collective was searching – in a cultural sense – for a different interface or set of operating instructions for the presentation and dissemination of information in the framework of the hardware of television and the software of visual media transmission systems.

The fact that, in 1970, the term software could still be transferred, in a metaphorical gesture, from computers to an entirely different domain – that of video art and the critique and utopian vision of the media in general – indicates that software has historically had a broader meaning than simply a computer program. The 1960s-1970s meaning of software was much vaster than the instructions to a processor because it had the exclusionary meaning of everything that is not the hardware. It was the statistician John Tukey who, in 1958, first used the word software to refer to all aspects of the computer which are not the "tubes, transistors, wires, tapes and the like."[825] As Nathan Ensmenger writes in *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*:

> Although the idea of software is central to our modern conception of the computer as a universal machine, defining exactly what software is – can be surprisingly difficult. Although Tukey clearly intended these other elements to include primarily computer code, by defining software in strictly negative terms – software was everything not ex-

> plicitly understood to be hardware – he left open the possibility of a broader understanding of software…
>
> In this sense, software is an ideal illustration of what the historians and sociologists of technology call a sociotechnical system: that is to say, a system in which machines, people, and processes are inextricably interconnected and interdependent. Software is perhaps the ultimate heterogeneous technology. It exists simultaneously as an idea, language, technology, and practice. Although intimately associated with the computer, it also clearly transcends it.[826]

The Random House Websters print dictionary of 1991 has two definitions for the word software. The first definition is the obvious one, from today's vantage point, of a computer program. The second definition is: "any material requiring the use of mechanical or electrical equipment, especially audiovisual material such as film, tapes, or records."[827] Today, in the year 2022, this second sense has all but completely disappeared. It is not mentioned in the English-language Wikipedia article on *software* nor in any of the many available online dictionaries. What was the cultural-linguistic context in 1970 of the founders of the journal of the Raindance Corporation choosing the name *Radical Software*? We know that Tukey marked the change in signification of *software* in 1958 from human operators to computer software code (computers in the late 1940s and 1950s necessitated – mostly female – programmers to manually configure cables and wires and flip switches). But what about the video material meaning? What is its history? Who first started using the word to mean that?

Was the term software so widely in use when talking about computers that it spilled over into people referring to audio-visual material like video as software? Or, in 1970–1974, was it the opposite, that the second definition as multimedia content was still more widespread than the computer sense? Only later did the computer meaning ascend and the audio-visual material meaning fade. Did the creators of *Radical Software* in 1970 have little awareness of the computer connotation? Or did they have some awareness of it but did not think that the reality of the two different meanings was significant? Or did they intentionally want to evoke the computer meaning to make things more interesting?

Architecting Better Social Media

Michel Foucault speaks of the arena of the micro-physics of power, the invention of a machine for the governance of diminutive things. There is a potential battle looming between surveillance by the algorithms, databases, and data acquisition equipment of the big corporations and the resistance of my own everyday life practices of enjoyments and freedoms, the tug-of-war between power and anti-power.

How do we accomplish this radical progressive transformation in the age of information and online social media? How do we realize the next step in what social media can become?

In new media theory, there is the idea of an endlessly reproducible object existing in an entirely digital and virtual – and therefore non-physical – space. In a way, this emphasis derives from the influence in cultural studies of Walter Benjamin's infinitely cited essay "The Work of Art in the Age of Its Technological Reproducibility."[828] Each digital object is believed to be created directly from the 0s and 1s which are held to underlie it, and which stand discretely for presence or absence. If we start out instead from higher programming languages, we see that the patterns of software coder and user experience are more complex and are genuinely material and architectural. The idea that computers can represent everything – the Alan Turing idea of the universal machine – leads us to miss out on all that can be architected.

What is Creative Coding?

What is Creative Coding? Everyone knows what computer science is and what programming and writing software code is all about: it is a technical discipline, an engineering subject, an established practice of learning about how we get something to run, how to write a program to do something for us without making an error. It is a purposive-rational activity, driven by objectives like making money, implementing a cool new application, or the aesthetic fascination of engineering and all technical details as ends in themselves. Technical universities train their students in computer programming. All businesses employ computer programmers: banks, insurance companies, car manufacturers, telecom providers – the list goes on and on. Every company maintains a huge database, transaction system, and IT know-how.

In the 1960s with video art, artists created artworks that explore the possibilities of technology and/or modify media to communicate aesthetic and socio-political concerns. These genres include new media art, digital art, electronic art, interactive art, generative art, software art, code art, Net.Art, VR art, robotics art, cyborg art, Bio Art, sound art, telepresence art, and ecosystems art.[829] In the past fifteen years, artists and designers have become increasingly interested in learning how to write software code. This trend has been fueled in part by specialized development environments for Creative Coding (special toolkits for artists and designers), such as Processing, openFrameworks, Cinder, Max/MSP, and vvvv.[830]

So far artists have only rarely questioned the conventional understanding of computer programming. It has been taken for granted that programming *is what it is*, and that Creative Coding is the decision that the list of categories of people who should learn how to program should expand. A whole new category of students is going to acquire those same skills which students at engineering schools acquire. The idea that the nature of programming will get changed by those involved in the humanities, design, art, and cultural studies is only now emerging. Creative Coding promises to break new ground for affecting the design patterns of culture. The aim is to create a hybrid discipline merging technology and the humanities.

Over the decades, computer programming has represented a series of successive and different paradigms and undergone revolutionary paradigmatic changes. These seemingly technical paradigms are in fact knowledge paradigms which are to be understood

as a genealogy or sequence of cultural-historical stages. We should view these successive phases in terms of cultural and historical knowledge. Indeed, it is no simple task to see, recognize, or define what computer science is! Computer scientists, who have been trained in a mono-disciplinary way as experts in technical practices, do not have any perspective on themselves.

The Poetic Expressiveness of Code

We want to examine the early twenty-first century culture of software code as advancing a poetic or expressive media, or which interrogates code as an artefact, or which develops code in relation to writing. Although software code is generally assumed to be a formal language allegedly lacking the ambiguities of human languages, there exist both subjective and anagrammatic sub-texts within code. These expanses of textuality are to be found both in explicitly Creative Code and in normal coding practices. They can be brought into relief through practice of the artist-programmer or via deconstructionist readings of standard code.

How can the ambiguities of language reassert themselves within a formal or logical language? The software layer is the translation between human and machine language. This traversal actuality of translation and corporeality of the human factor already make code to a certain degree – and potentially even more so in the future – sovereign from the hardware-level bit-manipulation functioning of the computer. There is a vast array of experimental projects.[831] There is experimentation with the rules of code and disobedience of the rules. Prose and poetry and the writing of fiction get integrated or interspersed with the code of various programming languages. Computational media engender new poetics. Computers become *writing machines* (N. Katherine Hayles) and *phantasmal media* (D. Fox Harrell).[832]

Code is not only an instrument of language. It spawns new language environments. Poetic language re-emerges within software code to counteract the original historical and scientific-technological axiomatic assumption that code is a series of instructions to the machine, an exercise in formal logic, and the conversion of language to information. What is the relation of software code to the history and future of writing? What is electronic writing and how are literary texts today (contemplating the inverse direction of the relationship) affected by the structures and idioms of informatics? Might Creative Coding develop into a challenge to the understanding of what programming is – a contestation and transformation of informatics by artists and cultural scientists with a commitment to the humanities and the arts and design?

From Sociology to Media Studies to the Next Paradigm?

Martin Cooper led the engineering team at Motorola that designed the first cellular portable phone prototype (the DYNamic Adaptive Total Area Coverage) in 1973.[833] It was ten years before Motorola's portable phone was made available to the public. Cooper says that he was inspired by the handheld communicators of *Star Trek* of the 1960s.[834] The

communicators on *Star Trek* are compact units with a flip-up transceiver antenna grid. Opening the flip-antenna portion activates the device, which one can then speak into without dialing. The MicroTAC was introduced by Motorola in 1989 as the world's first flip-phone design.

The smartphone as the exemplary technology of digitalization brings to our attention the exigency of defining a knowledge paradigm beyond those which take as their object of inquiry society or "the social" (sociology) and media (media studies/media theory). This new field should bring transdisciplinary knowledge to bear on the design of informatic technologies. We need a discipline of the aesthetics and morality of algorithms.

Sociologists believed in something the "the social." But this was wishful-thinking – the masses resisted being known or accounted for by the surveys and questionnaires of the market- and social researchers. This resistance takes the form of a hyper-conformism to the questioners' polls and expectations.[835] Sociology's idea that *the social* is an objective scientific reality is questionable. *The social* is a construct – as in Berger and Luckmann's "the social construction of reality" – yet the word social in their phrase is self-contradictory.[836]

Media studies was a promising and then anointed candidate to succeed sociology. The idea that *the media* is an objective scientific reality that will always be here is also questionable. With the smartphone, there is no longer the mediation between two "realities" nor McLuhan's extension of man. We are in a situation of interconnectivity that is global, all-encompassing, and viral. Information, messages, and other things we value spread through the networks because they are contagious or infectious for us. This propagation knows no boundaries and is promiscuous, as evidenced in phrases like "going viral" and "viral media." Media are everywhere. The coronavirus crisis was "real" and deadly, yet it serves as well as a metaphor for the borderless and replicating nature of the media.

The smartphone is a combination of many technologies, an assemblage (a concept of Deleuze and Guattari, Manual De Landa, and Bruno Latour) or apparatus (or *dispositif*, a concept of Giorgio Agamben).[837] The user seeks interaction with and mastery over the world through informatics. There are algorithmic automatic coded procedures. There is the combinatorial state-altering manipulation of systems-and-applications options and properties – the "settings." Media and "the social" still play a residual role. In the posthuman, we are now information processors designing our social-media-digital-virtual existence through software.

Thinking back to the 1970s and 1980s, Marxist-oriented sociologists continued to insist for a long time that economics and class relations (or antagonisms) between workers and capitalists in the sphere of production are the driving force or "determining instance" that explains society and the world. Marxists did not take seriously continental postmodernist thinkers like Jean Baudrillard and Umberto Eco, who prioritized the study of media, consumerism, cultural semiotics, and the power of images and rhetoric to destabilize modernist truths and core values like democracy, communication, and the public sphere. The golden age of sociology was the hegemony of the knowledge paradigm whose primary object of investigation was "the social" or society. Baudrillard deconstructed the epistemological model of "the social" in *In the Shadow of the Silent Majorities… or the End of the Social*.[838] After the social science resistance to media for decades, we are now in the

golden age of media studies. Today the size of media studies departments at universities dwarf sociology by an order of magnitude.

Philosophy, psychology, and literature are knowledge fields for understanding the *I existence*. For understanding *we existence*, the media have become the object of inquiry of the dominant knowledge paradigm in the social sciences, humanities, and art and design. But can media studies explain software? The intense emphasis on media is a resistance to a newer paradigm which is emerging, indicating an inflection point analogous to how sociology resisted the emergence of media studies and media theory in the 1970s and 1980s. This third paradigm now coming into view after sociology and media studies has to do with existence, experience, experiment, engagement, emotions, and embodiment. It has to do with code grasped and appreciated in a transdisciplinary way and with the importance of *posthuman* agents. It deals with the street art of *the construction of situations*. It addresses the relation between philosophical morality and computer science algorithms. It deals with post-scarcity post-work, and with pragmatic-utopian visions of a better society, and with *technological anarchism*.

John M. Culkin brought his "Center for Understanding Media" from Antioch College, Yellow Springs, Ohio to the New School for Social Research in New York City in 1975. Starting in the 1990s, media studies/media theory succeeded sociology. Are we now on the verge of the supersession of media by a newer paradigm of software studies, Critical Code Studies, or transdisciplinary informatics?

The ubiquitous digital media technology device of the smartphone is versatile. I can do anything with my smartphone at any time, and from anywhere my body physically finds itself. It knows so much about specific urban and geographical localities, and about the online or offline status of my "friends" at this instant. I micro-manage the environment of my smart home. I chat and text with others, peer-to-peer, many-to-many, or one-to-many, sometimes with avatars and AI bots in social networks and virtual worlds. I play games. I snap and browse photos. I photoshop-edit and upload my photos. I mashup videos. I read the news. I make my schedule of today's activities with the calendar app. I check the weather and the financial markets. I pay for things. I do my banking. I map my travel route. I order a taxi or Uber. I remote-control my car. I check in for my flight or train trip. I listen to music. I stream movies. I watch sports. I study a foreign language. I read an e-book. I monitor my health and my calorie intake. My smartphone doubles as a flashlight. BUT WHO IS THIS "I"?

A Happy Ending

What was this book about? The question was posed: how can cultural theory explain the effects of technology on society? There were six "answers" given or conclusions reached.

(1) We need to further develop a cultural theory concept of hyper-modernism that goes beyond the concepts of modernity and postmodernism.
(2) We need to further develop a cultural theory concept of hyperreality that goes beyond the concept of reality.

(3) We need a cultural theory concept of post-humanism that goes beyond the concept of humanism.
(4) Hyper-modernism, hyperreality, and post-humanism are nowadays implemented through software code.
(5) We need a way of thinking and writing about the world which I call "science fiction theory" or science fiction as an epistemological mode – beyond the received idea of science fiction as taking place only in the expressive fictional genres of novels and films.
(6) Following Marx's eleventh thesis on Feuerbach – "The philosophers have only interpreted the world; the point, however, is to change it" – cultural theory becomes *praxis*.[839] *Praxis* is the unity of theory and practice.

I argue that the practice of Creative Coding, informed by media theory and cultural theory (or transdisciplinary design), is the appropriate way forward for my work in the humanities (in art and media research). Creative Coding is also understood as a challenge to informatics, the possibility of inciting a paradigm shift in computer science itself.

Marxist thinkers place at the center of their vocabulary the term "capitalism." They constantly name the society in which we live as "capitalism," emphasizing economics rather than culture, and believing that to be the most insightful way to describe the world. They tend to make the unaware assumption that they are speaking of an "objective reality" rather than having selected a prism through which to view things, employing a concept which they have chosen. Paradoxically, I use the term "capitalism" myself many times in this book. Yet my primary standpoint is that the Marxist perspective is suspect because naming our society by its "economic system" of capitalism implies that there is a clear alternative when there is not. It indulges in the abstraction that there is a "something else" which is almost never explicated. The stress is regrettably almost always on critique of *what is* rather than the design of *what could be better*. The Marxist thinker Mark Fisher admirably glimpsed this conundrum in his 2009 book *Capitalist Realism: Is There No Alternative?* when he pointed out that it is henceforth easier to imagine the end of the world than to imagine the end of capitalism.[840]

There are many other central concepts which purport to name our society. The German sociology of Max Weber privileges the aspect of bureaucratization.[841] The French sociology of Émile Durkheim speaks of *anomie*, something akin to alienation.[842] Liberal thinkers might say that we are living in a liberal democracy. Conservative thinkers might call us a decadent society where standards of excellence have declined. Spokespersons for ethno-religious democracies such as Israel or Morocco might state that they live in a Jewish or Muslim society. Post-Frankfurt School critical theorist Jürgen Habermas underscores the positive gains of the Enlightenment and the program of "communicative rationality."[843] Feminism might say that we live in a patriarchy. Paul Virilio claims that we live in a permanent state of war or militarism. For Baudrillard, it is hyperreality or the simulacra. Post-humanist ecological thinkers point to our anthropocentrism. Christian thinker Jacques Ellul's famous book was *The Technological Society*.[844] Systems theorists like Luhmann and Nassehi accentuate *complexity*. Lyotard identified the post-modern society. Hannah Arendt and George Orwell decried totalitarianism.[845] Daniel Bell named the contemporary situation as the post-industrial society.[846] Others might say the infor-

mation society, the informatic society, or the reign of cybernetics. Michel Foucault highlights power and surveillance. My project is to add my idea that we live in a science fiction world to this knowledge list.

I invoke all these examples mainly to make the point that the Marxist emphasis on capitalism is not the only way of looking at things. The hyper-modern world is in dire straits, and we need to engage with all the above-enumerated ideas to find a way out. We will not find the solution merely by rearranging who has "ownership of the means of production."

My political and intellectual orientation is that I am on the far left but I am not a Marxist. I am also a sort of liberal and a sort of anarchist. I would like the intellectual left to become self-critical about its perennial Marxist assumptions. I would like to develop a full-fledged alternative theory and framework for socially transformative thinking. I have taken one step in that work in this book with my controversial claim that we are living in science fiction. To underline the paradigm, worldview, or epistemology of science fiction provides considerable understanding of what is happening out there and what we can do about it.

Another significant support for my thesis about Creative Coding comes from the philosophy and history of science. As Kuhn teaches, any given science evolves progressively and mutates in its history, proceeds through paradigm shifts and periods of "normal science," and is intricately bound to the cultural *Zeitgeist* though not in a relativist way. My many retrospectives of the technological history of digitalization offer evidence that programming has changed many times from paradigm to paradigm, and always in parallel with cultural paradigms. All this strengthens the idea that programming can change again.