Society of the Instance

by Alan Shapiro

The entire virtual life of societies in which postcapitalist conditions of seduction prevail presents itself as an immense accumulation of software instances. Everything that was previously either venerated or rejected in the fixated mirror stage by the ideologically constituted ego as fetish system of commodities, structurally sustaining panoply of consumer objects, or imposing spectacle of images has dissolved away into an uncharted virtual reality of the real time instantiation (or permutable, momentary actualization) of modeled and coded, flexible, extensible, beautiful software "objects." My investigation must therefore begin with the analysis of a software instance, and the process of instantiation into an (objectless) "object" (little unit of distributed artificial intelligence) of what is known in the theory and cultural practice of object-oriented programming as a software class.

Object technology today embraces everything from object-oriented languages and engineering methods, object-oriented databases, and distributed communications middleware for the Object Web (standardized CORBA / Java or Microsoft's Component Object Model) to the Unified Modeling Language project spearheaded by Rational Software Corp., integrated development environments for building Windows applications like Delphi and Visual C++, moving world 3D graphics authoring systems, and advanced MIT Media Lab research in AI, nano-wearable computing, autonomous agents, and artificial life. The new object-oriented technology, however, is not just a burgeoning movement for greater elegance and efficiency within the software industry. It is also an energetic tendency influencing and thoroughly reshaping all of globalizing, contemporary culture, as well as social and individual existence. The "postconsumerist" multimedia software instance - with its instantaneous (complementary to instantiating) hyperlinks; its intensification of visual, windowed information; and its multiple, effortlessly inter-translatable "tracks" all rendered from the same transmitted stream of digital bits (Negroponte, *Being Digital*) - is the emblematic software object of this current trend of affairs. Informatics must thus be viewed simultaneously from within as a neo-science or professional practice, and from without as the principal transformative cultural (and epistemological) force of our times.

Within the hyper-ascendent trajectory of the pragmatic employment of computers and software, object-orientation (along with other great innovations of the personal computer revolution, such as interactive graphics or PCs as communications media) has been a huge benefit and a source of vastly improved useability. As compared with the earlier imperative, functional, and procedural methodologies of computer science and software development, object-orientation, considered "from the inside" (from a vocational standpoint), plainly brings its practitioners into closer contact with the dynamic processes of the "real world." Considered "from the outside," however (from the standpoint of discerning the philosophical and cultural effects of object technology), the operation of carrying over the realist conception of a consensus-ordained, observable, external reality

from classical, natural science to computer systems - which are virtual, not physical - is something I regard as highly spurious and seek to place into question. Looked at from a "business" point of view, object-orientation clearly lifts professional programmers up to a loftier vantage point from which they can construct systems which are not only more robust and less costly, but which either more accurately emulate already existing non-virtual systems which they are intended to automate, or more prolifically implement new applications which have been inspired solely by the digital imagination and its creative tools.

Procedural programming combined the imperative (computer as executor of sequential instructions) and functional (computer as calculator of mathematical values) approaches into a unified technique whose advantage was its capacity to break down large, complex requirements or tasks into smaller, more manageable parts. The basic modular component of a classic procedural programming language like C, known as a "function," is both imperative and functional. In C, a function (equivalent to a procedure or function in Pascal) both carries out a succession of operations, and returns a computed value to its calling function. As a "problem-oriented" language, however, C remains a dialect(ic) of the (scientific) Enlightenment, a final, release compilation of the rationalist-empiricist epistème of Bacon, Newton, Descartes, and Leibniz.

Unlike its object-oriented successor C++, C is obliged to maintain an unyielding separation between data (data structures) and the computing functions which operate on that data. This is precisely because the designers of the C programming language (Kernighan and Ritchie), or of the Pascal programming language (Niklaus Wirth), were not able to think about the crucial problem of code reusability in any terms beyond the logic of binary opposition between the "subject" process of the executing thread (identified by projection with the computer scientist herself) and the already written, reusable pieces of code which are "not the subject." The subject thread temporarily relinquishes its control over program execution to reusable code modules which are conceived in the very image or reflection of the scientist-subject. These helper routines or function libraries are delegations or extensions of the scientist's "purposive-rational intelligence," her problem-solving, data-obtaining capabilities. The archetypal scientific subject empirically observes and analyzes the external, natural world with the aim of "acquiring data" about it, and then either generalizing towards the attainment of Enlightening knowledge or interpreting the data in the Light of verifiable or refutable theories and hypotheses. Data is a significant goal, and the computer is an essential tool which assists in its procurement.

Given the Cartesian epistème's stringent dichotomization with regard to things between the "is" and the "is not" (Leibniz's universal combinatorics of the applied binary code) or between the self-assured cogito and the self-evidence of the physical world, data could never be a constituent part of the fundamental building blocks of any representational or computational system dependent on this paradigm. Data could only be something which is "passed" back and forth among the system's core compositional units (the functions), or which is conceived as residing in input and output data structures which get reconciled in a supplementary "data mapping." The conceptualization of code and data as inseparable, integral aspects of an intrinsic, indispensable, cohesive entity called the "software object" necessitated the abandonment and going beyond of the

Weltanschauung of empirical, binary, subject-object-based, reality-promoting science.

At a certain indeterminate (Canetti) point in its history, science shifts from making use of a physical model of the universe to making use of a virtual model, from the "original" mission of "getting closer to" and upholding reality to adopting an agenda of strengthening and promoting virtuality. Such a major about-face in a knowledge system is only feasible, of course, because the effectuated transmutation is the outcome of a subtle, always already present, immanent reversibility. The unexpected flip is provoked by the teasing out of potentialities which were inherent within the system in its "original" configuration, and which new circumstances have summoned to the fore (in this case, the post-1968 social climate in the West which engendered the retooling of survival strategies of postcapitalist seductive power). In the empty, penumbral center of the momentous inversion of realist science into virtualist science stand the magisterial adventures of computer science, the emanation of the software instance, and the spreading, omnipresent machinery of (multimedia, windowed, information-concentrating) instantiation. The object-oriented (computer) scientist is no longer an observer and data analyst of the physical world, but is a designer and builder of anamorphic, virtual worlds.

From the perspective of the cultural and cognitive consequences of object technology (rather than from a strictly vocational perspective), the greater efficacy enabled by the new paradigm is a double movement (of reversibility) concurrently both closer to and away from "the real." Object-orientation's true merit is relative: it is a great thing for computers, whose serviceability it enhances considerably in comparison with what computers were before. Object-orientation is, however, not necessarily such a great thing for reality, since its spiraling appeal and potency cause more and more aspects of life to be brought daemonically under the sway of instantiating computers and the reign of the multimedia instance. The gesture "towards the real" within the double reverse movement of object-orientation is perceptibly self-contradictory and self-sabotaging, even though it appears at first glance only to be ambivalent. On the one hand, this directional vector "towards the real" is partly and frequently dressed up as a matter-of-fact claim about increased apperception and faithful representation of reality. It is a duplicitous gambit which continues to cash in on the still-paying dividends or rhetorical remainders of the self-evident "scientific real" left over from the very rationalist-empiricist epistème which the partner motion "away from the real" (exemplified in the concept of the "software class") seeks to subvert.

On the other hand, the distributed software object is partly and frequently lauded for its proximity to reality according to the ways of seeing and cheery virtues of a new (post)simulationist, neo-Platonist epistème. Presented within this fresh (Platonist) framework, the idea of the software object hinges directly on the metaphor of already familiar, existing physical and social objects. Object-orientation is an approach which appears to encourage the intellectual and artistic elaboration in the design process of cybernetic or quasi-structuralist "models of reality." What I wish to demonstrate is that the misperception of the twofold simultaneous movement "towards the real" and "away from the real" as an apparent ambivalence rather than a contradiction (hiding the possibility of immanent reversal) is due merely to muddled dereliction in sorting out the locations

and maneuvers of the two respective epistémes. The neo-Platonist move "away from the real" is rhetorically disguised, thanks to the legacy of scientific rationalism and its seductive force, as an element of concretion belonging to the move "towards the real." The postcapitalist system withholds its potential for liberatory, human-chosen, seminal reversibility by short-circuiting each of its vectors at a limit point into its opposite (the motion "towards the real" - towards the "play of appearances" of the revivified object - flips into virtuality; the motion "away from the real" flips into the modeling, cartographic, simulational, computer-assisted real). However, the double movement at the heart of this procedure remains self-contesting. At the same time, it both depends upon and progressively undermines rationalist-empiricism's residual assumption that reality always remains intact, no matter what new, pernicious cultural operation is brought to bear upon it.

Two of the key conceptual innovations of object-oriented methodology are the software class and the software object. The central notion of software class is defined as an abstraction of the common properties of like things. The class of trees, for example, is designed to encapsulate both the attributes and operations (data members and function members, in the terminology of the C++ programming language; fields and methods in the terminology of Java development) which concern all trees (or at least those trees which are available in a specific virtual worlds modeling environment). There may then be descendent subclasses of the tree class, such as the beech, myrtle, mahogany, pine, and cypress "families of trees" classes, and yet further descendent subclasses of these intermediate level classes. Each new derived subclass inherits characteristics from another class which is one abstraction level higher in the class hierarchy or chart, although these property inheritances remain largely theoretical until the critical moment of object instantiation. Fir, hemlock, larch, and spruce trees might be examples of still more specialized classes which derive from the pine tree family class. Redwood, incense cedar, and juniper tree classes might derive in turn from the cypress tree family class. A distributed software object which is instantiated as a "particular" redwood tree - in a virtual reality Redwood National (Theme) Park - might have some properties which are coded and maintained in the class from which the software object is immediately instantiated (the redwood tree class), other properties which are coded and maintained in a class which is one level higher in the class hierarchy (the cypress tree family class), and even further properties which are coded and maintained in the base class of the class hierarchy (the root nodal tree class). The base tree class, depending upon the given system implementation, may itself be descendent from an even more underlying and abstract base class with a name like CWnd (Microsoft Foundation Class Library) or TForm (Object Pascal or Delphi), denoting some fundamental element of the system, such as, respectively, a (Windows) window or a screen dialog backdrop. In actual simulation or modeling practice, interrelated groups of software classes are conceptualized diagrammatically and in multiple connections as part of a project-specific inheritance ranking of classes and descendent subclasses which are deemed to be suitable to the patterning and management of a distinct environment, platform, or application domain. Targets of such occupational, object-oriented, class diagram design might include a graphical user interface operating system, a bank of elevators in an office building, a four-dimensional texture mapped virtual world, or the client-server transactional operations of a large business enterprise.

In my arboreal hierarchy example, or tree (in the data structures or diagram sense) of trees, the base class Tree might have the attributes (data) of Color and TextLabel, and the operations (code) of Draw and MakeRustlingNoise. The intermediate level class CyprusFamilyTree might have the attributes of NumberOfCones and ConeType, and the operations of Shake and Rotate. The most differentiated class RedwoodTree might have the attributes of Height and TimberGrade, and the operations of Protect and ChopDown. If I, the applications programmer, then choose to create an instantiated software object and call it MyRedwoodTree, I can then set the attributes of this new software object (usually during, and as part of, the act of instantiation) to MyRedwoodTree.Color = BrownishRed, MyRedwoodTree.TextLabel = "Keep your hands off!," MyRedwoodTree.NumberOfCones = 300, MyRedwoodTree.ConeType = Leathery, MyRedwoodTree.Height = 200, and MyRedwoodTree.TimberGrade = Excellent. The result, in my virtual world, of instantiating the software object and setting these attribute values will be a redwood tree "instance" which is brownish red, has a text label which reads "Keep your hands off!," has 300 leathery cones, is 200 feet high, and seductively attracts crowds of virtual lumberjacks. Similarly, the software instance MyRedwoodTree can be programmatically manipulated by calling the operations MyRedwoodTree.Draw (graphically renders the tree), MyRedwoodTree.MakeRustlingNoise (plays a looping .WAV file), MyRedwoodTree.Shake (stirs the tree in a specific cyprus tree family manner), MyRedwoodTree.Rotate (turns the tree in a specific cyprus tree family manner), MyRedwoodTree.Protect (activates a killer force field around the virtual redwood tree), or MyRedwoodTree.ChopDown (sacrifices the redwood tree to the timber industry).

The point of exhibiting all this detail is to rigorously nurture my contention that the instantiated, distributed software object has achieved a state of existence which is finally beyond the logical Cartesian or classical, mathematical physics dualism between the "is" and the "is not" (or even beyond the discriminative categorization of the "this" and the "that" - the modernist, Saussurian, proto-linguistic system of arbitrary, positive differences among phonemes). A given instantiated software object both "is" and "is not" like another short-lived software object instantiated from the same abstract parent classes. The same particular transient software instance both "is" and "is not" like the specification of attributes and operations coded and maintained in each stratified, abstract class from which this software instance gathers its behavior and conjoins its evanescent appearance.

Beyond a certain (im-precise) point in time, without realizing it, object-orientation definitively transgressed the limits of the discrete, binary, nominalist, symbolic logic which was the "original" foundation of computing. The software instance, as the basic (de)compositional unit of this new (post)simulative system, enacts data- and context-specific performances of its ancestor classes, at last unifying data and the operations on that data into a single, self-contained entity. Initialized in real time, and in precise circumstances of seduction for each new occurrence, the distributed object coalesces its parameters of existence "on the fly" from coded and maintained, detailing and specifying, (de)constituting parts. Unlike binary bits, which were the elementary particles of earlier, classical computing, these latter-day class layered "elementary particles" are undecidable and non-discrete to the core. The exp ress characteristics conferred on the software object

at the moment of its virtual inception include the instance's attributes, operations, memory state, inter-object messaging or event protocols, and associative and aggregational relationships with other distributed objects. With object-orientation, the problem of code reusability has been rethought in an ingenious and much more pliable way. This very suppleness, however, is also the secret mechanism of seductive power in the newly upgraded, virtualizing cultural system.

The passing of messages and the responding to events are also among the most important aspects of the behavior of software objects in the object-oriented paradigm. The Windows operating system, for example, is often discussed in terms of its "message-driven architecture," and most Windows applications are structured according to an "event-driven technique." Rather than consisting of code which acts upon its surroundings and chooses when to act in conformance with a self-driven interior logic (the old procedural paradigm), the Windows program is a first-level "command target" or message dispatcher which receives messages from Windows (itself an instantiated cyber-environment), parses them, and passes them along to the "window procedure" or message map of the appropriate window object within the application. Client area, component windows are themselves instantiated from specialized window classes like 256-color bitmap buttons or multi-column list boxes. A message intended for a designated child window might be something like a forced repainting or user input from the keyboard. The program sits idle much of the time, does its processing when called upon, and then returns control back to multi-tasking Windows.

In the Unified Modeling Language, the message is the basic unit of inter-object communication, and a message can only be sent between two objects which have a named association. An event is a type of message announcing a noteworthy occurrence. A message almost always holds either data or control commands intended for a target software object, but it can also contain "metadata" furnishing, for example, network-demanded protocol information about itself. Each (non-message) software object in a system specifies and makes public its own distinctive messaging interface, thereby signaling to other potentially communicating software objects the handshaking procedures, obligatory precursor events, and methods of message transfer which it supports. The inter-object message is itself a kind of software object, since it is also instantiated from abstract (message) classes, themselves enumerated in the standardized UML metamodel as an elaborate hierarchy of specialized synchronous, asynchronous, periodic, episodic, and bursty inter-object message subclasses.(1) The networking transport (layer) of software objects has also become the site of a dense, painstaking formalization, with increasingly ornate systems of message formatting, packeting, routing, prioritizing, reliability, and reporting. Inter-object messaging (in a distributed architecture) allows a software object to invoke an operation belonging to another "fine-grained" software object residing elsewhere on the network, and even leads to the going beyond of the concept of boundaries between local and remote machines. As messaging becomes even more prominent in distributed object technologies (for example, in CORBA 3.0 with its new compatibility with Message-Oriented Middleware), it is important to keep in mind that message and transaction objects are varieties of instantiating software object, and that their studied refinement only adds further to the repetoire of virtualizing techniques available to multi-mediatized culture and its relentless endo-colonization of lived experience.

The seventeenth century mathematician and philosopher G.W. Leibniz is credited both with assembling a protocomputer arithmetic machine which performed multiplication and division as well as addition and subtraction, and with devising a new branch of mathematics in his essay *De Arte Combinatorica* (1666). Leibniz followed René Descartes in wanting to deduce a complete knowledge system starting from a few basic tabula rasa principles of certainty. Leibniz believed in a "universal character" or universal logical language which someday would be inferentially constructed step by step on the heels of the establishing of the correct first propositions. For Leibniz, the selection of the quintessential grounding axioms for the lingua franca system entailed the contriving of a few absolutely requisite representational symbols for the prime concepts, and a few absolutely requisite rules for combining these symbols. Once the general system was successfully set up, all existing or new scientific and cultural questions could then be solved, according to Leibniz, by invoking the dictum "let us calculate." This dream of applied mathematical certainty was reinvigorated and pursued anew in the mid-nineteenth century by the formal logician George Boole (the calculus of finite differences, the algebra of logical reasoning), and in the early twentieth century by logical positivist philosophers like Bertrand Russell (the logical conclusions of first principle theorems for all of mathematics, the logical conclusions of first principle atheism for all of human beliefs). Leibniz's vision of an unrestricted method of automatic ratiocination by calculation was then actualized in the mid-twentieth century invention of the high speed digital computer, which was first conceived in 1936 by Alan Turing and Emil Post (in separate descriptions of code-driven, finite state automata), and then built by John von Neumann and his University of Pennsylvania Moore School Group colleagues during and immediately after World War II.(2)

Since any specialized automaton (precursor of the software application) could be delineated with a finite set of binary instructions, argued Turing in his 1936 paper, "On Computable Numbers, with an Application to the Entscheidungsproblem," therefore a universal automaton (precursor of computer hardware) could be imagined which would exactly mimic the desired behavior of any specialized automaton simply by cycling through those same instructions.(3) In his book Turing's Man, J. David Bolter characterized the information processing technique of a Universal Turing Machine as the replacement of "discrete symbols one at a time according to a finite set of rules." This "original" logic of computing was firmly rooted in the dualism of the "is" and the "is not" (the long strings of binary digits or 0s and 1s, the perfect "existential" weight of the discrete identifiers). It still had rather strong ties to the old physical model of reality (the substantiality of numbers, the switching of registers and signals in both storage and processing), and to the "certainty" - or identity with itself - of the old scientific object.

Prior to the compelling appearance of "object technology" in recent decades, computers were deeply and intricately associated with the triumph and concretization of mathematical, symbolic logic. As recently as 1984, the computer science and Classics professor J. David Bolter was able to write - in apparent obliviousness to the rise of object-orientation - that "every computer program is the ... realization, the tangible proof, of a theorem in logic ... every programmer ... is a logician with a theorem to prove." If anything was certain concerning the status

of electronic digital thinking in the history of ideas, Bolter asserted, it was that the land of the CPU and the fetch-and-execute cycle is a kingdom from which God and religion are, without shadow of doubt, excluded. "The unification envisioned by Plato" - the ideal world of the Platonic Forms and Ideas, the "series of perfect patterns from which the imperfect objects of the material world" are derived - "has no counterpart in computerized thought."(4)

With the great shift in the software development paradigm which kicked into high gear in the nineties, from structured and procedural programming languages (Fortran, ALGOL, Pascal, C) to object-oriented languages (Smalltalk, Java, Delphi, C++), we have reversed from the Cartesian subjectivism of Turing (computer as machine to imitate the intelligence of the logician) and von Neumann (strict division between the "subject" of program commands and the operated-upon data) to the neo-Platonism of object-oriented luminaries like Rational Software's Grady Booch (the diagrammatic modeling language is the program code) and Xerox PARC's Smalltalk inventor Alan Kay (the de-sensualization of children's play on the computer screen depicted in glowing McLuhanesque terms as an "extension of man"). In an ironic twist of fate, Western thought goes back and slurps thirstily from the primal fount of Platonist metaphysics, returning full circle to the favoring of the abstract ideality of essences over the sentient reality of single case appearances, to the disseminated purity of the "transcendental signifiers," to the systemically deferred, yet urgently required, attendance of God.

The move against antinomies seems oddly to be refracted and paralleled throughout all of postconsumerist, mainstream cyberculture. The anti-Saussurian "raising of the bar" between paired terms (signifier / signified) has led "merely" to the decoupling of dualisms which turned out to belong to the time-bound, "doomed" Enlightenment epistéme. The assault on binary oppositions has cleared the way for newly emergent, brilliantly resourceful persistence strategies on the part of occidental metaphysics (postcapitalism). Object-orientation, with its class hierarchies and virtual object instantiation, appears to have introduced a "deconstructive Platonism" (Plato revised by grammatology), a vast "production" of interminable, subject-less "writing;" a layered, metonymic chain of (re)valuation; a perpetually and recursively auto-substituting, auto-supplementing, structured and seductive game of différance. The virtual tree which I see outside the three-dimensional digital video window (or screen) of my computerized car is no longer a real tree. It culls together its existence from the (differentiating and deferring) class hierarchy of trees. It both is and is not a tree, because its différance has been put into play.

Object-orientation's strategem of entity generalization and concomitant specification of instances is strikingly reminiscent of the Platonist Realm of Forms and Plato's accompanying critique of merely technical or representational copies which do not partake of the "Idea" of the original. Software instantiation (de)institutes a temporary relationship between an ordered ranking of software classes and the created, then destroyed, software object which is a parameter- and data-specified instance of those determining classes. In Platonist terms of iconic likeness, a software instance would be regarded as a legitimate resemblance rather than an illegitimate semblance or simulacrum, even though the distributed software object is, in a certain sense, inferior to the less tangible software class due to the former's transitoriness. The significant contrast would be between the

system of classes / objects and its predecessor, the classical "society of the spectacle" (or société de consommation) system of lowly, imitative images.

In the section of *The Republic* entitled "How Representation in Art is Related to Truth", Socrates sounds uncannily like a guru of object-oriented design when he says: "Let us take any common instance [!!]; there are beds and tables in the world - plenty of them. But there are only two ideas or forms of them - one the idea of a bed, the other of a table. And the maker of either of them makes a bed or he makes a table for our use, in accordance with the idea." Primary reality, for Plato, is not to be sought in the empirical world of everyday things (ordinary instances of beds or tables), but rather in the general, abstract Forms (the divine idea of the bed or table) from which "concrete" things are derived or fashioned. Socrates goes on in this passage to say that there are three philosophical categories of b eds: the idea of the bed (made by God), the instance of the bed (made by a carpenter), and the imitation of the bed (made by a painter). Concerning the question of how near or far each of the three categories of beds is to the Ideal Forms of Beauty, Truth, and Excellence, it is clear for Socrates that the idea of the bed is the closest to these exalted virtues, the instance of the bed comes in as a respectable second closest, and the imitation of the bed runs a pitiful last - far removed from anything valuated as either noble or good.

The Socratic dialogue in Politeia about mimesis is a contemptuous critique and dismissal of (imitative) poetry and painting, which only reproduce technical copies and are said to be "thrice removed from the truth." Painting, for Socrates-Plato, is a degraded art form of the semblance or mirror image, an aesthetic activity which demands of the painter "no knowledge worth mentioning," and no comprehension of "true existence." Moreover, although it "may deceive children or simple persons," imitative painting comes up way short in its endeavor to fool the majority of members of the polity into being placated by its inauthentic, second-rate images. Media technologies of mere duplication or representation are ultimately inadequate because they fail as instruments of political superintendence. Object-orientation is a new "artificial language" of seductive power (a new system of virtualization) which curiously combines a Platonist ingredient (transcendental signifiers or software classes) and a deconstructionist ingredient (unending, undecidable, auto-referentially recursive "discourse" in the hierarchical, (de)instituting, subclassing chains).

In an interesting excursus on différance, Canadian cultural theorist Gary Genosko isolates the disparity between Derrida's and Baudrillard's anti-semiologies or respective critiques of Saussure as located in a decisive difference in emphasis placed on the "orders" of either value or signification. In his examination of the Saussurian sign (in “Speech and Phenomena”, in *Of Grammatology*, and in the essay "Différance"), Derrida focused on the negative "linguistic concept of difference without positive terms," taking apart Saussure's (mistaken) dualistic metaphysics of signifier and signified, and thus leading to critical recognition of the impossibility of the sign's self-referential unity or full presence to itself. The doubled, horizontal relationship of value, which relies for its structuration on the sign's two internal components (signifier and signified) and the bar between them, is shown by Derrida to be a myth. By consequence, the vertical relationship of signification - the bar between the sign as a unified entity and what the sign excludes - must also be mythical. Signification, which depends for its existence on

the establishment or "institution" of a positive plenitude of the sign, is prematurely subsumed under value, which has already undergone the negative critique of the subversive, differential play of language. The edifice of signification is subordinated to the foundation of value.(5) Once the foundation fails, the building is - much too quickly - believed to collapse along with it. The rapid stress applied by Derrida to the bar of value both denies to signification the operational preconditions for its critical (and fatal) scrutiny, and underestimates the intractability of the auto-rectifying, re-complexing, continually morphing, positive sign. The negative critique of the transcendental signifiers is an auto-proliferating, contiguous, reiterative critique which bypasses the otherness of both subjects and objects. It turns one in a direction away from any enduring, ironic, sacrificial engagement with the eighteen-headed hydra of signification or the simulacra.

Object-oriented software engineering and multimedia design (in their prevalent forms) are languages for the streamlining, administration, control, and substitution of human experience. As a cybernetic, "artificial language" (human languages were always artificial, of course), object-orientation has profited a great deal from its in-depth familiarity with "so-called natural languages." It is keenly aware of the (negative) différance which Derrida in some sense claims to be a force or quality possessed by all languages which is subversive of metaphysics. In order to devise an "artificial language" of decentralized control or regulation of the quotidian, it helps to know a lot about the functioning of the "revaluation or dislocation chains" of human languages. Seductive power is also exercized all the more efficaciously after language has been transferred to an electronic, cyber arena, where it becomes more "programmable" than ever before. Software classes (the Platonist pole of the system), however, are "reconciled" signifiers (they are not, in turn, signifieds of yet more signifiers, and so on ad infinitum). They are the system's closure points, the limits to the free play of discourse, and the very stipulations for virtuality. After the menacing "subjective" unrest of 1968 (which continued well into the seventies), the subject-object system of "mere images" (Plato's simulacra), technologies of representation, and clear separation between viewer and viewed (or between consumer and artistic product) was judged to be insufficient for the preservation of the polis. The metaphysical tradition was then renovated and renewed.

In the early years, I was submerged by a governing spectacle or consumer society - a "system of objects" categorized collection which had not yet blossomed into a consummated system of distributed (temporally deferred and differentiated) objects, not yet become a language. Baudrillard paused for a long moment to contemplate the specific weight or unity of the socio-cultural sign in its methodical, signifying logic or "semiological reduction" of ambivalence, reciprocity, and sensuality. The sign, in its wholeness, was still structured in vintage consumer society according to a taxonomy of positive, "arbitrary" differences from other sign-objects (within the Saussurian, modernist, cybernetic-operational, Cartesian epistème). After this prolonged hesitation Baudrillard moved on to his own, more deliberate, long-term project of "deconstruction" or prying apart of the signified-signifier relation into the denominator of simulation models and the numerator of the (fourth-order) simulacra or pure, crystalline objects.

The early emphasis in Baudrillard was on the stability of the sign in its positive configuration, the ruling semiocracy, or the collective, autoerotic perversion and

"passion for the code" which institutes a serialized social "cohesiveness," and is opposed en bloc to the suppressed anthropological principle of symbolic exchange. "To become an object of consumption, an object must first become a sign ... it is thus arbitrary ... it derives its consistency, and hence its meaning, from an abstract and systematic relationship to all other sign-objects." In the vertical order of signification of the (not yet distributed) system of objects, difference is still organized on the level of undivided signs, in the bar of separation between the coded abstraction of the joined signifier / signified and that which this "ideological unity" radically exiles (Bataille's sumptuary expenditure or Mauss' potlatch and obligatory gift exchanges). In the self-endorsing mass media of "speech without response," what is sold to "consumers" with each advertising message or replicated image is not a particular product, but rather the system of abundance and diaphanous communication in its entirety, the order of signs itself. "Without 'believing' in the product, therefore, we believe in the advertising that tries to get us to believe in it."(6)

The special and technical speed-effects of the broadcasting electronic medium overwhelm the content or real events which the medium is naively believed to beneficently bring into relief or report on. This inversion of message and medium brings about the most non-obvious, yet most insidious, kind of manufacturing of pseudo-event. "Substitution of a 'neo-real' for 'the real' is occurring everywhere, produced as a whole based on the combination of the elements of the code. In the wide spectrum of daily life, an immense simulation process is taking place, in the image of the 'simulation models' on which the operational and cybernetic sciences are working." The generalized code - now understood in the sense of a flexible re-combinatorics of the broken down, unraveled, most elementary particles or units of something real, as well as in the "anti-semiological" sense - takes the place of the fading signified or referent (in its classic form).(7) In L'*Échange symbolique et la mort* (1976), this insight into the workings of what Baudrillard calls the "third-order simulacra" develops into a full-scale commentary on "the metaphysics of the code" - the obsession with genetic information (DNA) and its affiliated, "micro-molecular" command and control sequence-transcription dialects as the new definitional matrices of life and reality.

What opened the door for the notion of simulation was the pressure which Baudrillard applied to the signified (or referent), in the first of two moves (the second being the move from the signifier to the fourth-order simulacra of potentially reversible, "accursed share" objects) leading to the dismantling of the "doubly uneven" sign. Baudrillard pulls the thread of simulation from the sign-spindle's fragile, tapered end (the signified), and keeps on pulling. Since capitalism relies so much on the instrumentalizing equivalence of exchange-values and on the dynamic, charismatic allure of continually mutating signifiers, use-values and signifieds recline into a reduced ideological status as orbiting satellites, justifying alibis, or retrograde "security deposits" (des cautions) of the former terms (exchange-values and signifiers). Needs are posited as "rational" and "natural" finalities by the consumer system, but they are, in fact, insatiable "desiring symptoms," tautologically projected by the "personalizing" signification and selling process itself. Reality and lived experience (le vécu) lose their standing as self-evident, objective references, and are substituted by "reality-effects" and "live-effects" which the signifiers still require for collateral purposes of legitimation and fresh data input. The "structural revolution of value," as it is called in *L'Échange*

symbolique et la mort, has completely unhinged the sign from its referent. It has left the emasculated signified in a state of relative enfeeblement whose apprehension is still partly obscured by both the rote trumpeting of Enlightenment virtues and by the downright celebration of simulation.

The reconstitution of a thing based only on its information is just what the doctor ordered for a society bent on cultivating its ability to allow individuals to live alone in ostensible safety, while simulating a life of rich experiences (interactions with the enchanting, technical twins of other people and the world). History (which was perhaps always a simulation model), for example, is shattered to bits by the celluloid, tape recorder, and document stockpiling apperception of events (*L'illusion de la fin*). From within a culture of simulation, as is our situation, claims of cognitive or hermeneutical access to antecedent "real history" can no longer be verified or sustained. Memories of the Vietnam War are replaced by memories of Hollywood movies about the Vietnam War. Referential substance and "truth" are everywhere replaced by a superabundance of information and data. Ordinary reality is replaced by the permanent buzz of "entertainment" - the eternally recurrent, stereotypical hyperreality of television and of all the Disneylands.

Simulation of the body in plastic surgery, bodybuilding, cosmetics, and "beam me up, Scotty." Simulation of thinking in artificial intelligence. Simulation of sexuality and desire in pornography. The end of war in the Pentagon's video game virtual war machine. The end of linear time in the no-time of real time, and in the retroactivity and turbulence of time travel. The end of aesthetic illusion in the holodeck, and in neural-direct and helmet-and-glove VR systems. The end of communication in the over-proximity and ubiquitous connections of telecommunications. The end of fiction (and its opposite, reality) in "science fiction" films which only publicize and disseminate the newly arriving digital technologies and their prescribed living conditions. The end of the referent of human beings in the will to build a technical, immortal replacement species. These simulation systems are assemblages belonging to the "third-order simulacra" - mongrel, duplex contraptions where the signified (catalyzed by a dose of "fatal theory") has absorbed all of the energy of its signifier into itself, thus generating a parodic and bloated "exorbitant" version of what it already was. But compartmentalized American thought sees nothing to be troubled by in any of this. Plastic surgery is a signifier of the body or of social success. An inner ear cellular phone implant is a signifier of connectivity. A nano-cartridge memory implant is a signifier of my freedom to lie (to myself). Campbell's Soup cans are a signifier of soup - and that's that.

This "Californian" hyper-pragmatism of the bi-structured sign is a single-minded lucidity of the technological agent and its purposive ends. Its ideological resiliency serves as a poignant reminder of the prestigious and prodigious ground still shared between the third-order, cybernetic, combinatorial, "mode of information" codes and Enlightenment-rationalist science. Considered from an "anti-sociology of knowledge" point of view, simulation systems still suffer from the preeminent malady of the long era of the classical scientific epistème: the prevailing illusion that the world is not an illusion, or the epochal nullification of what Baudrillard calls "the vital illusion, the radical illusion of the world."(8) Science has always insisted on an unwavering identity of the object with the object - on the pinning down of a fixed, determined object in its material (or informational, transcribed)

constancy. This predilection of science has effectively concealed the absence of things from themselves, the fact of their differentiated and temporally deferred "incarnations," the fact of their non-immediacy, and the seductive play of their appearances.

So-called "primitive" societies, as Baudrillard and others have often pointed out, had a different (more symbolic, reciprocal, and creative) relation to the problem of the truth or reality of this world - a question which Western society resolved through technical simulation. With instantiation and the software instance - with a certain artistic practice which they invite - the opportunity to recover (or reverse) what Paul Virilio calls "the aesthetics of disappearance" comes into view. It is the excess of information, inaugurated by the postcapitalist cultural system itself, which provokes the "objectlessness" of the object, the apparition of the object which is not an object: the software object. Here lies the possibility of a fully original anthropological moment (the flip of the software instance into the Nietzschean play of appearances), and the potential contestation of the scientific worldview.

The "interface" of a human to the world (a relationship which should respect that which makes the world, or an object, indifferent towards me) - this contact area of illusion or "fiction" is, properly speaking, the territory of the signifiers. After the signifieds have been cleared out of the way (they imploded as simulation systems), Baudrillard is free to undertake his second move in the strategic divestiture of semiology - a step which will complete the supplanting of the object as structural sign by the object as pure sign. This is a creative and proactive - yet "non-subjective" - involvement with new "signifiers." In a gesture which is at once political (though without attracting power) and (post)aesthetic, Baudrillard returns to his original passion for (the system of) objects, but this time in a meditation on "pure objects" (as well as "pure events" and "pure images"). These new "evil genius" objects, born of the system's exorcistic and excessive processes, unknowable through their information, shaped in their superficial and empty molds as "unconditional simulacra," are the carriers of irony and shrewd indifference. Images were denigrated and expelled from the "body politic" by Plato, and they are again today - their singularity rolled over like subliminal cannon fodder in high-speed, windowed, multi-component instantiations. The dizzying qualities of these object-oriented instances start to resemble the "aleatory giddiness" which Roger Caillois, in *Les jeux et les hommes* (1958), saw as characteristic of the ethnographically archetypal game forms of chance and vertigo which, as with gambling, are now making their comeback.

I identify the post-1968 dominant cultural strategy of object-oriented cyberspace as the construction of a new signification system, erected on the scaffolding of the previous one. As far as mainstream cyberculture is concerned, this transition from the third- to the fourth-order simulacra can be summed up by the formula: Simulation plus Différance equals Virtuality. The deconstructionist boxer struck a forceful blow against the Cartesian epistéme. This arching jab turned out, however, merely to be one of the best video replay highlights for the recombinant resource database, to be used in the prize fight promoter's overall scheme for the differentiated distribution of the spectacle onto every screen and holographic living room table in town.

Plato's metaphysics of reality proclaimed a tripartite structure, posed on the equilibrium between (for example) the divine idea of the table (the transcendental signifier) and the supposed "real" exemplification of the table produced by the carpenter (the transcendental signified), to the double exclusion of the image created by the painter. The stability of this system has already been shaking for quite some time. The software classes of object-oriented programming are a symptom of this loss of steadiness. Software classes are relatives both of the "reconciled" signifiers and of the "reconciled" signifieds (related to the positive structuralist-semiotic sign), but they are also one step further along in simulation. With the progressive withering of referents, the "rock solid" instances of the carpenter's workbench evanesce into the conceptual abstractions (based on design and memory) which will generate virtual software instances. The software classes are the post-simulationist remains of (references to) reality, and they become a basic unit of the new virtualizing system. Transcendental signifier of the tree class, declared by the programmer, (de)coupled with the transcendental signified of trees "in reality" which he remembers, on whose memory this system survives, but which will not be around for long.

To accomplish the system of virtuality which is starting to reign around the globe today, the signifieds (the references to "reality") must also be dispersed, distributed, put through the work of differentiation, phonemically and temporally processed by what is alternatively called either linguistic deconstruction or the software subclassing hierarchies. The differential move is the limitless, extreme, free play of fourth-order simulacra - an entropic, friction-free, fractal "chaos" of (re)valuation. Différance, however, is only half of the equation's left side. The paradoxical, hybrid system (simulation plus différance equals virtuality) is simultaneously postcapitalism's temporary salvation and its fast burnout (its desperate reform). In spite of its extraordinary charisma and seductive powers, instantiation is in a much more precarious position than any of its predecessors in the history of capitalism (alienation, reification, simulation). The software instance relies on transcendental signifieds (trees) which will soon disappear, and on transcendental signifiers (the idea of trees) which will quickly follow into extinction. It also relies on the differential qualities of language, which it will soon exhaust as well.

Notes

1 - Bruce Powel Douglass, *Real-Time UML: Developing Efficient Objects for Embedded Systems* (Reading, Mass.: Addison-Wesley, 1998); pp.25,56-61.

2 - Herman H. Goldstine, *The Computer: from Pascal to von Neumann* (Princeton University Press, 1972).

3 - *The Computer: from Pascal to von Neumann*; pp.274-75.

4 - J. David Bolter, *Turing's Man: Western Culture in the Computer Age* (Chapel Hill: University of North Carolina Press, 1984); pp.22, 47, 77-79.

5 - Gary Genosko, *Baudrillard and Signs: Signification Ablaze* (London: Routledge, 1994); pp. 18-24.

6 - Jean Baudrillard, *The System of Objects* (translated by James Benedict) (London: Verso, 1996) (originally published by Éditions Gallimard in 1968); pp.166,200.

7 - Baudrillard, *La Société de consommation: ses mythes, ses structures* (Paris: DenoÎl, 1970); pp.194-96.

8 - Baudrillard, *The Perfect Crime* (translated by Chris Turner) (London: Verso, 1996).