A particular relationship between the icon on the desktop and the programming language

Keywords:
Icon, Allegory, Interface, PYGMALION, Smalltalk

Masanori MIZUNO, Nagoya University

Abstract
This paper examines a relationship between the icon on the desktop and the programming language, in order to examine the icon's significance in the Graphical User Interface.

In order to explore this relationship, my paper regards the icon as a pictorial sign and the programming language as a linguistic sign. I can then consider this relationship as a special case of battle between the pictorial sign and the linguistic sign throughout cultural history. I then focus on David. C. Smith's doctoral thesis PYGMALION, which introduced the icon concept for the computer, and the programming language for PYGMALION, Smalltalk, in order to investigate what the icon brings to the computer and why the icon is even possible. Next, I analyze why the linguistic sign becomes the pictorial sign from the viewpoint of analogy and allegory. Finally, my paper shows what is concealed by the desktop icon.

Through these analyses, we can say that the relationship between the icon and the programming language is one of allegory and complementariness. This means that the icon is a pictorial sign, restricted metaphorically by the linguistic sign at the computer program.

This paper is the English translation of the original Japanese paper which is published in Journal of Socio-Information Studies, Vol.10, No.1 and awarded for the Best Paper of the Year 2005 for Young Researchers.
1. Introduction

In the information society, almost all of us look at the computer display which shows the surface called the 'Desktop'. There are many icons on the Desktop and we select and click them with the mouse everyday. However, we don't give consideration to what the icon is on the computer display. Therefore, Barbara Maria Stafford writes the following about the icon: "At the close of the twentieth century, it should give us pause that we still lack a flexible method for orchestrating the jumble of discrete emissions and darting blips that swim across countless monitors. They remain a hermetic system of graphic symbols for which we have lost the analogical key (Stafford, 1999: 55). "It is time to find the analogical key. To that end, this paper examines the icon's significance not from the viewpoint of Information science but from the relationship between pictorial and linguistic signs.

In order to explore this relationship, my paper regards the icon as the pictorial sign and the programming language as the linguistic sign. We can then consider this relationship as a special case of battle between the pictorial sign and the linguistic sign throughout cultural history. Next, I focus on David C. Smith's doctoral thesis, PYGMALION, which introduced the icon concept for the computer. We look at the programming language for PYGMALION, Smalltalk, in order to investigate what the icon brings to the computer and why the icon is even possible. I make reference to Leibniz because Alan Key, the advocate of Smalltalk, thinks that Smalltalk has much in common with the monads of Leibniz (Kay, 1993: 70). Moreover, we analyze why the linguistic sign becomes the pictorial sign from viewpoint of analogy and allegory. Finally, this paper shows what is concealed by the icon on the desktop.

2. The icon as pictorial sign and the programming language as linguistic sign

M. J. T. Mitchell points out that "the history of culture is in part the story of a protracted struggle for dominance between pictorial and linguistic signs, each claiming for itself certain proprietary rights on a "nature" to which only it has access (Mitchell, 1986: 43)." From Mitchell's view, we can think of the computer as having both signs: the icon as the pictorial sign and the programming language as the linguistic sign. In the computer, the programming language generates the icon, but we can not actually look at the programming language, but only at the icon on the display of Graphical User Interface (GUI). Which of both these modes signs is dominant in the computer? We consider that the computer display with GUI is a contemporary struggle field for dominance between the pictorial and linguistic signs.

In order to investigate this relationship, I focus on the word "nature" in Mitchell's text. What kinds of "nature" do the icon and the programming language are approaching? The icon has brought the idea of "easy to use" into the computer world. Consequently we can say that the icon allows the computer user to recognize the situation of the computer "naturally". What about the programming language? The programming language is one of many artificial language which people have made for recognizing and communicating...
Masanori Mizuno—A particular relationship between the icon on the desktop and the programming language

special information efficiently, like numerical formulae or a chemical formulae. Therefore, the programming language is also a tool to facilitate our recognition. However, it has a unique characteristic which no other artificial languages have. According to Levine & Reingold, the computer is the first machine able to follow linguistic instructions (Levine & Reingold, 1987: xvi). Therefore the programming language has to communicate not only to people but also to the computer. Haruki examines the programming language from the viewpoint of communication of human thought, and writes its characteristics as follows:

The programming language as an artificial language sends an instruction to the Central Processing Unit (CPU). If we view this action from a different angle, we can think of it as a conversation between the human and CPU. With other artificial languages it is supposed that humans communicate with humans, but with programming language, it must be presumed that the CPU is the other half of the communication partnership. Moreover, CPU can understand only von Neumann logic and never tries to learn other languages. Therefore, the first purpose of the programming language is to be able to communicate between the CPU and humans (Haruki, 1989: 30-31).

Von Neumann logic is the basic architecture of the computer which has never changed since the beginning of the computer. It is a design model that uses a processing unit and a single separate memory to hold both instructions and data. In von Neumann architecture, "a structure of the main memory ability is linear and processes a basic task sequentially based on a program counter in the CPU. Therefore, we have to make a sequential process with data corresponding to the memory unit and task corresponding with the processing unit if we want to write a problem as a computer program (Haruki, 1995: 203)." From this computer principle, the linguistic sign has absolute dominance over the computer because the linear structure of language is suitable for controlling the CPU sequentially.

However, D. C. Smith tried to control the CPU with icons in the computer display in order for the pictorial sign to gain dominance on the CPU. Smith's challenge introduced a metaphor which shows a link between the real world and the computer world. After that, the spread of GUI which controls the computer with icons has created "a metaphoric space mediated by information technology" (Ishida, 2003: 332) onto the computer display. It represents a revolution from linear programming language to the icon as a non-linear information display. Therefore, GUI has signaled the end of the absolute dominance of the linguistic sign and initiated the beginning of the dominance of pictorial sign in the computer. However, it is important to consider that the metaphorical space on the computer display is created from the programming language which has an unique property. This change shows that the pictorial sign and the linguistic sign continue fighting each other for proprietary rights on a "nature" to which only each of them can access in the new stage, the computer. For that reason, it is useful to consider the relationship between
the icon and the programming language from the viewpoint of the pictorial sign and the linguistic sign in order to make clear what the icon is on the computer display. For that purpose in the next section, I refer to D. C. Smith's PYGMALION and consider what has been the impact on the computer of the introduction of the icon as pictorial sign.

3. Complementarity of the icon and the programming language

![Figure1. Pygmalion on Macintosh (Smith, 1993: 31)](image)

In 1975, D. C. Smith introduced the icon into the computer world with his doctoral thesis "PYGMALION: A creative programming environment" and now we take for granted that there are icons on the computer display. In his thesis, Smith defined the icon as follows:

Communication between user and computer is through a primitive visual entity called an "icon". Icons possess several visual and mechanical attributes. The system maps the visual characteristics of icons into corresponding machine semantics. Icons subsume the notions of "variable", "reference", "data structure", "function" and "picture" (Smith, 1975: iv).

From his definition, we can understand that the icon is not only the pictorial form of information but also the connection with the linear structure of data of the computer memory. I will consider why Smith adopted the pictorial sign as the information display form for the computer. When he made a complete volte-face from the linguistic sign to the pictorial sign, his greatest purpose might have been to introduce the non-linear information display principle into the computer because he referred to the difference between the word and the picture again and again in PYGMALION as follows:

In other words, the relationship between pictorial elements has as much to do with spatially-derived meaning as with the elements themselves. But "linear" programming languages have no spatial structure at all. (By "linear" is meant a verbal language such as English, constituting of a sequence of words.) One of the few characteristics of "linear" languages that even approach spatial organization is indentation:

```plaintext
IF ...... THEN
    WHILE ........... DO
        BEGIN
            .
        END
    ELSE
```
At best this is only an indirect indicator of meaning. The vast potential of multidimensional communication is simply not realized in linear languages because they are spatially unstructured (Smith, 1975: 12-13).

In short, Smith compared the linearity of the word with the spatiality of the picture and argued that the pictorial sign was much better for information display than the linguistic sign. Further, he explained the relationship between the linguistic sign and the pictorial sign using the words "Fregean" and "Analigical". Smith quoted these two words from Aaron Sloman's thesis which subtitled "the role of intuition and non-logical reasoning in intelligence". Sloman wrote that analogical representations such as maps or models were much more useful for "intuition and non-logical reasoning" than the "Fregean" mode of representation, named after logician and philosopher, Friedrich Ludwig Gottlob Frege, that was utilitarian tool suited to our reasonable logic (Sloman, 1971). From Sloman's idea, we can say that the alphabet is most Fregean mode of representation because it is at a high level of abstraction of the object being expressed.

As already noted, the computer is a logical machine which has linguistic ability. Therefore the instructions to the computer must be logical. As a result, the computer program is composed of linguistic sign which can think in abstractions and depict things logically. J. D. Bolter insists that the computer generates new space for the linguistic sign and writes about the relationship between the logic and the computer program as follows:

Computer programming is simply the newest version of symbol manipulation that mathematicians and logicians have practiced for centuries. Programming is embodied logic: the establishment of logical relationships among symbols that are embodied in and empowered by the memory chips and processors of the digital computer. Mathematics has been a special kind of writing at least since the evolution of modern notation in the 17th century. The set of mathematical equations that defines a physical theory is a symbolic text of the highest order. And science itself has been a formal language since the time of Descartes and Leibniz, or indeed Galileo with his claim that the book of nature was written in the language of mathematics. In the 19th and 20th centuries, the desire to make language formal and rigorous has led to modern symbolic logic, to semiotics, to logical positivism, and ultimately to computer programming (Bolter, 1991: 9-10).

Even though the computer programming is the latest version of logical operation that has made the linguistic sign more formal and rigorous, Smith dared to introduce into computer programming the pictorial sign which is much more useful for "intuition and non-logical reasoning". He wanted to adopt a way of intuitive description for computer programming. The pictorial description allows us more creative programming beyond logic so that what is written can be understood at first sight (Smith, 1975).

Although the dispute as to whether human thinking is based on the word or the image is
still not settled, Smith considered that human thought was generated from the image because he regarded Rudolf Arnheim's "visual Thinking" (Arnheim, 1969) as important. Therefore, Smith concluded that the linguistic sign for the programming language was too rigorous for the programmer. Also, there was a gap between the user's visual thinking and the computer using the linguistic sign as information display. The gap had to be closed by the programmer or the computer. From the birth of the computer, the programmer had to convert his/her thought into formal and rigorous linear language in order to close the gap. However, as the computer developed the gap grew too big to fill. Smith's PYGMALION introduced into computer programming the pictorial sign a form analogous to human thinking, thus closing the gap and making the program more natural for the user.

Using the pictorial sign as the icon, PYGMALION brought a visual map into the computer display to control the linear data processing of the computer. This map released the user from the rigorous linear language which was difficult to understand at first glance. Icons, however, are much easier for us to understand because they give visual clues of their own information. On the other hand, one can say the pictorial sign is forced on the computer in order to help the information display coincide with the way of human thinking since the computer has its own very different principle of linear data processing. How could Smith do that? Michel Serres gives a hint as to the answer. He says that there are two orders in the mathematical language which is the most formal and rigorous artificial language:

As for me, I cannot understand B without advance knowledge of A. Conversely, I need B and its continuation in order to understand A. However, we must make sure what the irreversibility is here because the irreversibility does not belong to the mathematical essence, strictly speaking. If anything, there are two mathematical orders. One is to find a solution, so it is irreversible. As we know, people go from unknown to known, from simple to complex and from easy to difficult. Although this is the way to discovery, this is not the order of the mathematical world. It is the order of the mathematician's practice. In fact, the order of mathematical world is limitless reversibility. Even though we cannot say that everything is irreversible, there are many ways to one concept or idea. Leibniz, a philosopher with multiple view points and an ambiguous system, knows this (Serres, 1968a = 141).

Serres divides the mathematics into two parts; one is the mathematical world which has reversibility. It is based on Leibniz. The other is the mathematician's practice which is irreversible because it has to find a solution. In the computer with von Neumann architecture, linear order is superimposed on the order of the mathematician's practice. However, Ivan Sutherland who developed the first computer graphic system: Sketchpad points out that "The sequential approach to mathematics is not required inside a computer, but the mathematical approach we normally take to problems does not encourage us to think of approaches other than sequential ones for the solution of problems. Nearly all computers in operation today
perform individual steps on individual items of data one after another in time sequence (Sutherland & Mead, 1977: 210). "In short, although the mathematical approach and the characteristic of the computer make the information processing in the computer sequential, linear and the irreversible, it is not an essential property for the computer because it is just "the order of the mathematician's practice" as Serres said and "The sequential approach to mathematics is not required inside a computer" as Sutherland said.

Well, mathematics as an artificial language is linear and processes data step-by-step. This is especially true of algebra. However, mathematics has another field, geometry, which deals with the pictorial sign. In mathematics, geometry and algebra are one aspect of the struggle between the pictorial sign and the linguistic sign. Mitchell writes about the relationship between geometry and algebra as follows:

The other analogy which offers itself is the relationship between algebra and geometry, the one working by arbitrary phonetic signs read progressively, the other displaying equally arbitrary figures in space. The attraction of this analogy is that it looks rather like the relation of word and image in an illustrated text, and the relation between the two modes is a complex one of mutual translation, interpretation, illustration, and embellishment. The problem with the analogy is that it is too perfect: it seems to hold out an impossible ideal of systematic, rulegoverned translation between word and image. Sometimes an impossible idea can be useful, however, as long as we recognize its impossibility. The advantage of the mathematical model is that it suggests the interpretive and representational complementarity of word and image, the way in which the understanding of one seems inevitable to appeal to the other (Mitchell, 1986: 44-45).

In this quotation, Mitchell shows that algebra which is solved linearly is translated into geometry which is solved by arranging arbitrary figures in space and vice versa. Although he remarks on the impossibility of the too perfect analogy, he concludes that algebra and geometry need each other in order to understand themselves. Therefore it suggests to us "the interpretive and representational complementarity of word and image." From Mitchell's idea, we may consider that the icon gives us a way to escape from abstract and linear programming language composed of strict logical statements because Smith displays the icon in the computer screen in order to apply the complementarity of the linguistic sign and the pictorial sign. This idea of the complementarity of the icon and the programming language reminds us of Serres' statement that "we must think things by arranging them in an expressive space instead of making a line (Serres, 1968b: 31) " in order to understand the system of Leibniz. Furthermore, this makes it possible to think that we can control the linear processing of von Neumann logic not by the linear language, but by a matrix composed of putting the pictorial signs on the surface one after another, in other words, "putting arrangements of multiple lines alternately
(Serres, 1968b: 31). "Therefore, Smith could introduce the new concept for the computer world, the icon, which connects with functions of the computer in order to control its linear information processing, not by the linguistic sign, but the pictorial sign.

However, Haruki points out that "all things are described progressively in the world of the von Neumann computer, so that non-progressiveness is excluded (Haruki, 1995: 222-223). "Moreover, Serres writes that "a line is a set and a sequence law at the same time (Serres, 1968b: 32). "Serres's comment is consistent with the von Neumann architecture computer which processes data linearly. As these statements suggest, we have to consider whether the complementarity of the linguistic sign and the pictorial sign comes into effect in principle in the computer world. Then, why did Smith introduce the icon into the computer against the linearity of the computer world? In order to make this question clear, we need to examine Alan Kay's Smalltalk, an Object-oriented programming language, because it is what made PYGMALION possible.

4. "The order of co-existences" generated from Information hiding

Smalltalk, proposed by Alan Kay in Xerox Palo Alto Research Center (PARC), is the first object-oriented programming language. Kay described Smalltalk characteristics as follows:

Smalltalk's design--and existence--is due to the insight that everything we can describe can be represented by the recursive composition of a single kind of behavioral building block that hides its combination of state and process inside itself and can be dealt with only through the exchange of messages. Philosophically, Smalltalk's objects have much in common with the monads of Leibniz and the notions of 20th century physics and biology (Kay, 1993: 512-513).

Moreover, Levine and Rheingold wrote that Smalltalk did not improve old programming languages, but rather introduced a new idea into the computer programming:

Smalltalk was one of the first and most spectacular creations of the generation of programmers who had grown up using interactive programming, ever-faster transistorized components, and ever-growing memory capacities rather than the old batch-processed, tube-based, limited-capacity computers of the FORTRAN and COBOL era. With Smalltalk came a new metaphor of computation as a system of software objects, all containing their own data and instructions and carrying on computations by exchanging messages rather than by performing instructions. Smalltalk was more than another computer language. It was a portal to whole new way to think about what computation is and what it can do (Levine & Rheingold, 1987: 216-217).

Here, we want to consider why Smalltalk could present such a new way for computer programming? Haruki suggests a principle of Information hiding which means that information inside an object is hidden from other objects, and objects cannot interfere with each other. This is an important idea for
Masanori MIZUNO—A particular relationship between the icon on the desktop and the programming language Smalltalk:

Although the object is just a structure which hides its own inside information from other objects on implementation, this mechanism of Information hiding has a special effect. We can say that almost all advantages of object-oriented are offered by Information hiding. Due to the hiding of inside information of implementation from the other objects, we can deal with the object as "entity" and as implementation which has successive memory space in memory units separately at the same time (Haruki, 1986 : 60).

As already stated, the von Neumann computer can process units of information only but simultaneously and it's memory is a linear structure. However, Smalltalk uses Information hiding in order to write a computer program without awareness of the linear information processing. In short, Smalltalk adopted negative action "hiding" in order to divide information processing into the linear one and the non-linear one. This distinction of the information processing makes it possible to show non-linear element, "picture" as an interface layer to the user while at the same time hiding linear elements, "variables", "reference", "data structure", "function" as a CPU layer from the user. Therefore, Information hiding creates new space in which many objects as information entity exist simultaneously without a linear hierarchy system and can send messages to each other. However, we have to examine what kind of order system this new space is without the linear hierarchy, which is generated from information. In order to make it clear, we refer to Leibniz's idea of space because Kay believes that Smalltalk's objects have much in common with the monads of Leibniz:

As for my own opinion, I have said more than once that I hold space to be something merely relative, as is time: that I hold it to be an order of co-existences as time is an order of successions. For space denotes, in terms of possibility, an order of things which exist at the same time, considered as existing together, without inquiring into their particular manner of existing. And when many things are seen together, one perceives that order of things among themselves. (Leibniz, 1715 = 1969: 682)

Leibniz called space "an order of co-existences." What does it mean? Yoneyama provides a useful guide to this question when he points out that Leibniz called music "Hidden arithmetic", the word "Hidden" being the most important for understanding the music:

Leibniz grasped that music was subordinated to arithmetic, and argued that the pleasure of music, or a complex representation such as enjoyment generated from consonance and dissonance existed in "Hidden arithmetic." It is very interesting and attaching the word "Hidden" to arithmetic is important. In order to understand its importance, let us put it aside for the present. In short, we say "the joy of the music is the arithmetic." Namely, we are aware of the arithmetic. In fact, we cannot enjoy the music immediately when we are aware of the
arithmetic in the music, for example, by counting or analyzing rhythm or chord structure. These are already other things from what we feel in the rhythm or beauty of chords. (Yoneyama, 1999: 198).

Although many scholars like Leibniz and Yoneyama consider that music and mathematics are similar, it is important that the music hides its arithmetic process in order to give us pleasure. Yoneyama points this out from Leibniz's expression, "Hidden arithmetic." Now, we return from this view of music to Leibniz's idea that the space is "an order of co-existences". Music is not just built of musical notes. Music appears when it hides its own arithmetic process, adding musical notes, in time, and we enjoy the "order of things among themselves." Therefore, we can say that Leibniz's "order of co-existences" is a relationship of many things which are result from the moment the process of arithmetic is hidden to us.

From the above, we can conclude that the non-linear information processing in the von Neumann computer put into practice by Kay's Smalltalk, and it's own principle of Information hiding are deeply very similar to Leibniz's "order of co-existences" generated from hiding its time. In conclusion, to hide the arithmetic process based on time generates "an order of co-existences" as space from absolute order, CPU's linear information processing in the idea of object-oriented. And, "an order of co-existences" gives the von Neumann computer an ideal space for mapping with pictorial signs, namely, the icons. Therefore, we can say that this is what makes for Smith's icons, to control CPU on the computer display.

5. An allegorical relationship between the icon and the computer program

As in previous sections, we make clear that the icon has a strong connection with the program which also has a connection with CPU's linear information processing. In such a connection, Smalltalk actualizes non-linear information processing by Information hiding and generates new space with "an order of co-existences" in the computer world. As a result, the pictorial signer, icon, can be placed on the computer display. However, we don't know yet why the linguistic sign --- the program in the layer of CPU--- becomes the pictorial sign as the icon in the layer of interface. In order to explain it, we now investigate next the relationship between the icon and the program from the viewpoint of analogy and allegory.

As already explained, the programming language communicates not only with the user but also the CPU. Because the programming language had become an incomprehensible one for us, Smith tried to create a new environment for conversation between the computer and the human being in the computer world by transforming the incomprehensible linguistic sign to the pictorial sign-the icon. And, based on the idea of "superimposing the world in the machine onto the world in the human being metaphorically (Nishigaki, 1994:70)," the icon became something which had to resemble something in the real world. After that, the computer display became "the desktop" because the linear information processing in
Masanori MIZUNO—A particular relationship between the icon on the desktop and the programming language consider what the analogy is, because it creates the metaphorical relationship. Leibniz wrote about when explaining the analogy in his "What is an idea?":

An analogy is said to express a thing in which there are relations [habitudines] which correspond to the relations of the thing expressed. But there are various kinds of expression. For example, the model of a machine expresses the machine itself, the projective delineation on a plane expresses a solid, speech expresses thoughts and truths, characters express numbers, and an algebraic equation expresses a circle or some other figure. What is common to all these expressions is that we can pass from a consideration of the relations in the expression to knowledge of the corresponding properties of the thing expressed. Hence it is clearly not necessary for that which expresses to be similar to the thing expressed, if only a certain analogy is maintained between the relations. (Leibniz, 1678 = 1969 : 208).

In Leibniz's idea for the analogy, we want to focus on his point that "it is clearly not necessary for that which expresses to be similar to the thing expressed." Because of a separation of the thing and the thing expressed, Leibniz shows us that it possible to consider that there is no resemblance what it represents in the analogical relationship. It is not important for the analogy to resemble what it represents in appearance but it must maintain some correspondence which shows they are the same thing. Likewise we can suppose that there is an analogical relationship if the icon maintains a
corresponding relationship with the programming language. Moreover, the computer programming language can make the corresponding analogy with its own logic. This is a large difference from the language of humans. In "A theory of computer semiotics," P. B. Andersen explains this as follows:

_The model system is generated from the system description_. Since there may be system descriptions that are not read by any system generator, system descriptions can occur without model systems, whereas model systems must be described by a system description. Therefore, model systems presuppose system descriptions. This assumption motivates a negative answer to our thematic question, "are there sound analogies between computers and humans" for the following reason: if program executions correspond to language usage, there is a difference between the system and usage concepts of informatics and linguistics, since in informatics, usage presupposes a system description, whereas in the linguistic variant I use, usage is of course possible without system descriptions: computers cannot do anything without descriptions, but people can. (Andersen, 1990: 125) [Emphasis is in the original]

In short, the relationship described by the computer programming language is an absolute order for the logic world which the human gives to the computer. It means that we can decide what a word in the programming language indicates at will and must annul the arbitrariness of language. As a result, the programming language is given some possibility for directly-opposed ideas; defining a rigorous meaning of word and making infinite links of analogy.

From these two possibilities of the programming language, we want to say that the linear program language pursues the strict meaning of a word and the object-oriented program language is open to infinite links of analogy. The reason is that, although the former is an analytical language system which works by defining a word meaning rigorously in order to describe the logic of CPU, the latter works by making links because the objects in the object-oriented programming language hide their own definitions which define the object itself, show their functional appearances, and generate analogical links between each other based on their appearances. However, the CPU's powerful von Neumann logic outside the language interferes with the programming language system in the computer. In terms of this point, the linear programming language and the object-oriented language are the same condition. Therefore, the computer program must be written for processing exactly, based on the von Neumann logic. Namely, the computer programming language keep to describe its own relationship by itself, but it is not a free relationship because CPU compels the programming language to make its own relationship which works on CPU.

Here, we consider that linguistic signs make their own absolute relationship by themselves in the computer world, and the object-oriented programming language generates the analogical links. Therefore, the analogical links have their own absolute relationship in the computer. Next, we refer to the work of
Masanori MIZUNO—A particular relationship between the icon on the desktop and the programming language

Angus Fletcher in order to make clear what the analogical relationship with the absolute relationship means. He writes "Let us suppose then that the proper analogue to allegory is the compulsive syndrome (Fletcher, 1964 : 286) '" he points out further that the linguistic sign tightly connects with the visual image in the allegory and that "Allegory perhaps has a 'reality' of its own, but it is certainly not of the sort that operates in our perceptions of the physical world. It has an idealizing consistency of thematic content, because, in spite of the visual absurdity of much allegorical imagery, the relations between ideas are under strong logical control. (Fletcher, 1964: 105) " In his other book, Fletcher examines the connection between the linguistic sign and the pictorial sign as follows:

Essentially a means of structuring language so as to produce continuously linked series of double or multiple meanings, this symbolic mode depends largely upon syncretic mixtures of symbols from which it builds up "levels of meaning," sometimes as few as two, or as many as seven. Minimally it holds that no single literal meaning can stand alone, but that a valid utterance must possess a transcendent meaning as well, a symbolic surplus beyond the literal level. Most allegories are images of cosmic order, and their fixed, hierarchical, and timeless character becomes problematic whenever such cosmic orders are subjected to temporal analysis. The key to the permanence of allegory throughout history appears to be its ornamental surface, which allies it with changes in cosmology and decorum and gives it an exploratory as well as a traditional and conservationist function (Fletcher, 1968: 41).

As Fletcher mentioned, the allegory includes a symbolic surplus which is generated from connecting multiple meanings beyond the meaning of the literal level with the logic outside the words. Moreover, the symbolic surplus transforms itself into the pictorial sign. The linear programming language is designed for correspondence with the outside logic, the CPU. Consequently, the language must be interpreted in its literal meaning in the computer. On the other hand, Smalltalk's information processing with object and message produces a different meanings system from the CPU due to the mechanism of Information hiding. This is writing various notes on the literal meaning indicated by the linear information processing. Furthermore, while the various meanings are tied up with the literal meaning, the linguistic sign forms an allegory which includes a symbolic surplus. In the computer world, this symbolic surplus becomes the icon on the computer display.

However, the icon on the computer display is not released perfectly from the literal meaning of the CPU. As already mentioned several times, it is just hidden by the mechanism of Information hiding in the object-oriented programming language. As a result, the icon does not require the user to interpret the literal meaning like the linear programming language but does require that we read itself based on certain rules. As for the limitations of the allegory, Fletcher points out that "since allegorical works present an aesthetic surface, this implies an authoritative, thematic, "correct" reading, which attempts to eliminate other possible readings. They
deliberately restrict the freedom of the reader. (Fletcher, 1964: 305) "Therefore, the operations possible with the icon are very restricted; "point out" or "move". This shows us that the pictorial sign, the icon, is not essential for the information processing in the computer but just a part of surplus. Even though the icon is a restricted pictorial sign, it has, however, some symbolic meaning beyond the literal meaning, and releases us from having to pursue the literal linear meaning for the CPU.

6. The allegorical relationship between the program and the icon, which the icon hides from the computer display

As mentioned in previous sections, the icon on the computer display and the programming language behind it show us two things. First, the object-oriented programming language hides the process of the CPU's linear information processing in order to give the user other meanings beyond the literal interpretation of the linguistic sign. As a result, it allows that some possibility of allegorical reading and summons the icon which has symbolical meaning to appear in the computer display against the strict logic of CPU. Second, although it is a property of the pictorial sign to escape the limits of a linear reading of words and to prompt a free interpretation, the icon is always limited in its interpretation because it needs the allegorical relationship with the linguistic sign in order to be generated in the computer display. We can reinterpret the computer display as the place where the complementary relationship between the linguistic sign and the pictorial sign unfolds. However, this complementarity is hidden from the computer display by the icon itself which has an analogical relationship with the real world. Consequently, we begin to believe that it is easy to understand and control the computer with the icon, and as we do not consider what the actually icon is. In this section, we want to examine how and why the icon hides the complementarity relationship between the linguistic sign and the pictorial sign from the computer display.

Figure 2. Dipintura, the original frontispiece for Vico's The New Science, 1744 (Fletcher, 1991: 148)
If Vico begins *The New Science* with an engraved frontispiece or "Table," The *Dipintura*, he also at once provides an extended verbal *Explanation of the Picture Placed as Frontispiece to Serve as Introduction of the Work" --- twenty-three pages in the English text. The Explanation shows verbally that the *Dipintura* pictorially reduces the twenty-three pages and by extension the whole of *The New Science* to a single complex image. In so doing the *Dipintura* doubles a temporally extended verbal construct, by synchronous emblematic means. Picture schematizes text, apparently without subversion (Fletcher, 1991: 153). [Emphasis is in the original]

This text by Fletcher is highly suggestive for us because he tells us that rather long verbal explanation is reduced to an image while it explains itself. It is a process by which the temporally extended verbal construct, which is always built in a reading act, is compressed into a pictorial sign which has simultaneity. In this process, the meanings of the linguistic sign are disposed in the order of co-existences generated from the pictorial sign. Then, the temporal construction of the linguistic sign hides itself from us and becomes a structure for a spatial construction in order to make and support a new pictorial sign. Therefore, even though the *Dipintura* is useful in helping us understand it more deeply, we should consider that we have to both look at its picture and also read its long explanation in the linguistic sign. Fletcher also asks a question; "Can a poet imagine a sequence that is devoid of the passing of time, when the poem shifts from space to space, point to point, as long as there are more than two points in the diagram (Fletcher, 1991: 173)?

Because there is a rather long explanation, the indications of the pictorial sign are arranged into a time line again and we can understand it. If there is no such explanation, the meaning of the pictorial sign drifts forever. The reason is that it is a property of the pictorial sign to be interpreted freely. Therefore, we have to demand of the linguistic sign a text in order to fix the meaning of the pictorial sign, although the verbal explanation forces us to interpret itself literally. This means that time hidden by the pictorial sign begins to move again with the linguistic sign which represents logic because "Time and the Logos are, no doubt, violently yoked together (Fletcher, 1991: 174)." As a result, we always end on a note of temporal description whenever we begin to analyze the order of co-existences of the pictorial sign spatially.

Now, we consider the icon from the idea above of Fletcher. As already mentioned, the icon is the pictorial sign which the symbolic surplus generated from the programming language change into. Therefore, the icon on the computer display must be reduced to a linguistic sign which is tightly tied to the time. However, Information hiding, the character of the object-oriented language which is fitted for making the icon, always hides its linguistic part. In other words, Information hiding always hides a part of the linear information processing from us. Due to its structure of Information structure, the icon is always displayed as the object hiding its basic part which is derived from the linguistic sign. Consequently, we can say that the icon on the computer display is an unstable pictorial sign.
because the meaning of it isn't fixed by the temporally extended verbal construct. This unstableness opens the icon to free interpretation, which emphasizes only the analogical relationship between the icon and the real world. And more, we just see the icon as a pictorial sign that can be freely interpreted and think that we can easily and freely control the computer with it. As a result, we don't look at the icon as symbolic surplus of the linguistic sign. It means that the allegorical relationship between the icon and the programming language, which restricts the icon's meaning, is made invisible.

7. Conclusion

We have considered the property of the icon in the computer display from the viewpoint of the pictorial sign and the linguistic sign. First of all, the icon is the pictorial sign, limited in its interpretation because of its allegorical relationship with the programming language which is restricted by von Neumann logic, the linear information processing. On the other hand, due to its structure of Information structure, the icon can't reduce itself to a linguistic sign in order to fix its meaning and is revealed to us as an unstable object which presses us to interpret it freely. As a result, the allegorical and complementary relationship between the icon, the pictorial sign, and the programming language, the linguistic sign, is made invisible in the computer display. From this study of the icon, we tried to clarified that the icon is symbolic surplus from the linguistic sign and is in allegorical relationship with the programming language. In the future, I hope to examine the relationship between the user and the icon from the allegorical viewpoint because there is a strong linguistic relationship between the user and the icon and this relationship limits the possibility of the computer interface.

Acknowledgment

We would like to offer our deepest thanks to the Hori Information Science Promotion Foundation for their support in conducting this research.

References


Haruki, Y. (1989) : Obujekto siko heno syoutai, Keigaku syuppan, Tokyo


Kay, A. (1993): The early history of Smalltalk, proceedings of 2nd ACM SIGPLAN History of Programming Languages Conference. ACM
Masanori MIZUNO—A particular relationship between the icon on the desktop and the programming language


