Marc Böhlen

#DEFINE

2003

https://doi.org/10.25969/mediarep/17616

Veröffentlichungsversion / published version
Zeitschriftenartikel / journal article

Empfohlene Zitierung / Suggested Citation:
Böhlen, Marc: #DEFINE. In: *Dichtung Digital. Journal für Kunst und Kultur digitaler Medien*. Nr. 29, Jg. 5 (2003), Nr. 3, S. 1–4. DOI: https://doi.org/10.25969/mediarep/17616.

#DEFINE

By Marc Böhlen

No. 29 – 2003

Abstract

Computers represent world through data and data types. The creation of data type reflects both the need for computational efficiency as well as the ideology of the engineers and scientists behind the code. The essay argues that the work of amateurs and artists can be seen as a contribution towards questioning and expanding the limitations of reality representation defined by computational requirements.

Writing computer code has become popular. Long gone are the days when only an elite class of specially trained individuals could squeeze colored pixels from clumsy machines. The advancement and global distribution of computer technologies in both hardware and software since the launch of first generation personal computers has created a fermentation ground for amateurs to engage coding and computation on their own terms. Every now and then the consequences of this become apparent: an email virus here, a hacked music file there and the occasional clandestine defacing of a corporate website. The fact that computers can be programmed and deployed out of their original context has not gone unnoticed by artists. For over three decades, artists have been building systems that manipulate efficient computing machines to control pleasure, disturbance and contemplation. This short essay will try to argue why the work of the hacker and that of the artist are meaningful contributions to the advancement of computing in general.

In 1976, Niklaus Wirth published "Algorithms + Data Structures = Programs"[1]. This classic work is one of the most insightful texts on programming ever written. The text is geared to the professional programmer interested in understanding fundamental concepts of data representation, from data primitives to files. What makes the text interesting even today is the fact that Wirth places the act of coding into a larger context, that of mental abstraction. The modernist Wirth formula for programs is elegant and simple; algorithms and data structures equate to programs. In the introduction, Wirth expands this formula to include the idea of abstraction. On the first page Wirth writes:

> "In all these cases (applications of storing and accessing data), the large amount of information that is to be processed in some sense represents an *abstraction* of a part of the real world. The information that is available to the computer consists of a selected set of data about the real world, namely, that set which is considered relevant to the problem at hand, that set from which it is believed that the desired results can be derived. That data represent an abstraction of reality in the sense that certain properties and characteristics of the real objects are ignored because they are peripheral and irrelevant to the particular problem..."

Wirth acknowledges what one might intuitively guess. The choices of data representation are meaningful. Furthermore, these abstraction choices are not universal and not objective. The model or representation of data will change with the need to express and make operational particular features of it. But what drives such choices? Beyond apparent necessity, it is a mesh of perception, thought and personal experience. The Sapir-Whorf Hypothesis, by which language forms the basis for thought, is out of favor after Pinker[2] and cognitive psychology made a clear case for an instinctual, biological relationship between language and thought. However strongly one might feel about hard linguistic determinism, it is difficult to overlook the fact that different human languages divide the world up into different compartments and emphasize differing features of it. Put this into a dynamic system in which many people communicate many ideas about many things, and one can assume that these different compartments have return values. In short, the language mind debate has a cultural context that is not as clear-cut as the biological based debate. In this sense, languages in dynamic cultural contexts influence thought. If language has this capacity, then tools that can be used to build languages do so as well. Computer code is such a tool. Data representation, therefore, is dependent on the limitations of code and the cultural fabric of language and thought in which the maker of code lives.

Computer code has a dual existence as symbolic representation and textual notation. As symbolic representation, it is interface to cluttered binary encodings of values and relationships, and a means by which to manipulate them. Because it uses the same symbols that written language uses, it can be read. Variable names are place holders for data and names for things. They contain choices on how to represent entities in time and space. Logic relationships reflect an understanding of how entities interact in time and space. This means two things. First, one can expect to find an intrinsic formulation of world view in every piece of code. Second, one can actively use computer code to formulate, within limits, how things might be imagined. Coding is the act of engaging in formalized representation tools that encourage constructions supported by the computational machine and refuse others. The limitations and the possibilities are equally significant. Just as the human voice tract has a limited frequency domain in which to form sound,

computer code is bound to the limitations imposed by the machine. Only that which can find a digital representation can be expressed by the computer. And within the boundaries of supported possibilities there are choices to make. The choices of abstraction and perception of relevance prune the problem domain to a task-particular subset.

Wirth acknowledges the role of selective abstraction but says nothing about the subjective nature of such selective abstraction. Are these choices not subject to the same mental and socio-cultural forces that guide other decision processes? This is where the making of code as language becomes subjective and personal. A telling example of this process is Wirth's illustration of the data type 'record', a data type comprised of a number of data primitives, often used to describe conglomerate features of an object. Employee or customer data is often modeled in a record-like structure, for example. Wirth uses this particular representation:

```
type Person =
        record name, firstname: alfa;
        birthdate:Date;
        marstatus: (single, married, widowed, divorced);
        case sex: (male, female) of
        male: (weight: real;
        bearded: Boolean);
        female: (size: array[1..3] of integer)
        end
```

Wirth describes this pseudo code as follows:

> "An example is that of the type Person, ..., in which relevant characteristics to be record in a file depend on the sex of the person. For example, for a male, his weight and whether or not he is bearded may be regarded as relevant in a particular situation, but for a female, three characteristic measurements may be taken as significant (whereas her weight may be confidential) ...[3]"

In Wirth's world, men have beards and three-way coded curvy women hide their weight. With unintended innuendo, Wirth's data construction reflects his perception of the world of men and women. In order to preserve the differences between the categories male and female, Wirth falls into clichés. Clarity has a price. In order to be efficiently operational, data structures need conciseness. Conciseness can reduce a complex entity such as a human being into a bland schema.

The argument made for low level data representation can similarly be made for logic and meaning representation: it can be done in more ways than one. Moreover, not all computer languages are equally suited for all purposes. While appropriate systems have been designed to work with numerical data, the problem of representing common sense knowledge, for example, is subject of intense and debated research within the computer science community[4]. The important point is that computer professionals have a very specialized and at times narrow view of how to represent the world. The results of their work reflect a small subset of how we can imagine our surroundings and the data within it. However, the results from these constructions have an increasingly pervasive influence on our lives. From employee databases to psychology profiles, we are subject to insufficient digital representations of human nature. The deep penetration of computing systems into society brings with it strong coloring of the entities the machines describe. The choices made to represent information define how this information will be used and understood. In this sense representation is information.

With the dispersion of coding tools to the masses, differing abstraction choices will be investigated. Data types, classification procedures and knowledge representation may be augmented to allow for additional conceptions of reality. But it is a slow and experimental process. However, the amateurs are becoming more sophisticated, the public more interested and computing professionals more attuned. We need to re-imagine the role of the computer and rethink how it can be programmed to visit less traveled paths. Chipping away at the monolith body of computer science, the hacker and the artist are both inscribing new desires into computing systems. In this sense their work is a contribution to the cultural diversity of the lofty universal machine.

Notes

1. Wirth, Niklaus, "Algorithms + Data Structures = Programs", Prentice-Hall, 1976
2. Pinker, Steven, *The Language Instinct*, HarperCollins, 1994
3. Wirth, p.21
4. Lenat, D. B. "Steps to Sharing Knowledge." In *Toward Very Large Knowledge Bases*, edited by N.J.I. Mars. IOS Press, 1995. See also: http://www.cyc.com/