CODeDOC II

Christiane Paul

As part of the CODE exhibition accompanying this year's festival, Ars Electronica invited me to curate a second installment of the online exhibition *CODeDOC* that I originally organized for the Whitney Museum of American Art's *artport,* a website designed as a portal to netart. *CODeDOC,* which launched in September 2002, was conceived to explore the relationship between the underlying code of software art and its results. A dozen software artists were invited to code a specific assignment—"connect and move three points in space"—in a language of their choice (Java, C, Visual Basic, Lingo, Perl) and were asked to exchange the code with each other for comments. The presentation strategy of *CODeDOC* deliberately deviates from the ways in which viewers usually experience a piece of software art, which commonly presents itself to the audience as executed code—the results of written instructions. In *CODeDOC,* the viewing experience is closer to the artist's creation process: what the audience encounters first is a page with the written code, from which they can launch its executed results. Since the assignment imposed substantial restrictions in format and file size, the contributed projects can't necessarily be seen as fully developed works; rather, they are comparable to small studies and sketches that capture an artist's approach.

Many of the prominent international practitioners in the field of software art could not participate in the first version of *CODeDOC* since the Whitney Museum is, by its mission, devoted to American artists (citizens and artists living and working in the US). *CODeDOC* presents a welcome opportunity to close that gap and widen the scope of the project. The eight artists/teams who were invited to contribute to the second installment and code the assignment—Ed Burton, epidemiC, Graham Harwood, Jaromil, Annja Krautgasser & Rainer Mandl, Joan Leandre, Antoine Schmitt and John F. Simon, Jr.—are mostly non-American. Some other artists who would have been obvious candidates for this project were not invited because they were already involved in other parts of the Ars Electronica Festival or exhibition. My special thanks go to Andreas Broeckmann for his input and suggestions in the selection process of the artists.

From its inception, *CODeDOC* was intended as a process-oriented experiment rather than an exhibition meant to make a specific statement or offer a certain point of view. Ideally, I wanted to raise questions about software art as artistic practice, and neither the outcome nor the reception of this project were easily predictable for me. One intent of the project certainly was to demystify the notion of code as a "mysterious," hidden driving force and to reveal the code to the viewer. Among the questions that seemed important to address or clarify were the following: does the term software art itself describe a certain form of aesthetics? Do "signature," "voice," and aesthetics of an artist manifest themselves equally in the written code and its executed results? Will reading the source code enhance the perception of the work? Does it in fact add anything at all or just create an emphasis on "technicalities" that is unnecessary, alienating, and obscures the work? How exactly could one define the relationship between the back end of code and its results?

The attempt to provide detailed answers to all these questions would be beyond the scope

of this introduction, and I just want to make some general comments and leave it up to the *CODeDOC II* projects themselves, as well as the discussions surrounding them, to offer further perspectives on these issues.
If one explores the body of work that each of the *CODeDOC II* participants has created over the years, it seems obvious that the label software art is a lowest common denominator for a formal description of their artistic practice rather than a term that describes specific aesthetics. The artists' works themselves cover a broad spectrum of individual approaches. The works of epidemiC, for example—which include *AntiMafia,* a Windows-based program for the co-ordination of associative actions, as well as the infamous *biennale.py* virus created for the 49th Venice Biennale (in collaboration with 0100101110101101.ORG)—are focused more on activism and the notion of software as cultural production. Ed Burton's *Sodaplay* and *Sodaconstructor,* which in the meantime have achieved cult status, explore the conceptual possibilities of "handcrafted" virtual robots as well as masses and their kinetic energy. Grahame Harwood's work has ranged from "pure" Perl poetry to software creation and narrative projects, such as the CD-ROM *Rehearsal of Memory*—which creates its interface out of a collage of the skins of the inmates and staff of Ashworth Hospital Authority—and the Web project *Uncomfortable Proximity,* commissioned by the Tate Museum, which reproduced the Tate website's layout, logos, and design, to tell a "different" history of the British art system. Compared to the former examples, Antoine Schmitt's works are far more visually oriented studies of the "behaviours" of forms in time and space.
While one might assume that an artist's approach (and perhaps even "personality") will manifest itself equally in the written code and its results, the code itself will naturally be more meaningful to other programmers than a general audience that might only get the roughest idea of its "mechanisms." Whether the code adds to an understanding of the work also varies substantially from case to case. One might speculate that the emphasis that the artists themselves would put on the importance of their code partly depends on the nature of their respective work: for example, artists whose work focuses on "raw" code (such as many of Graham Harwood's pieces) might consider the "written part" of the project more important than artists whose work is an exploration of visual forms, space, and action (such as many of Antoine Schmitt's projects). The presentation format of *CODeDOC* also seems to have imposed some (unintended) editing on the artists' part: in their comments, both Antoine Schmitt *(CODeDOC II)* and Camille Utterback *(CODeDOC I)* admitted that they felt compelled to clean up their code before presenting it to the public ("I'm one of those people that clean my bathroom if my friends are coming over," as Camille put it). One of the inherent dangers and certainly unintended effects of *CODeDOC* could be the misassumption that the quality of software art can be judged according to virtuosity and craftsmanship in the programming of code (that is, by criteria such as correctness, maintainability, lucidity, and readability, which were outlined by Donald Knuth). One of the beauties of art, no matter what form and material it takes, consists in the fact that its success is the result of multiple factors that cannot be objectively defined. A viewer could certainly enjoy the works of Leonardo da Vinci or Picasso on the basis of their outstanding virtuosity and craftsmanship alone (although they have much more to offer), but applying these standards to Duchamp's urinal or Beuys' "fat and felt" sculptures will presumably not yield relevant results or major appreciation. Like any other art form, software art cannot and should not be reduced to technical criteria, and the code should be seen as more than simply the wheels and gears driving the machine.
As an artistic medium and practice, software art seems to distinguish itself from other art forms such as painting, sculpture or film/video. As opposed to other forms of visual art, software artists write verbal instructions for their work that can be executed and produce

anything from visuals to a more abstract communication process (although the execution of code still requires various steps of interpretation and compiling and the code itself may be mostly a notation of logic). There is a peculiar relationship between the mostly hidden backend of code—which constitutes a convergence of language and mathematics—and the multi-sensory "display" it can produce: an "identity" in the sense of a sameness in different instances (code results), each of which takes a very different form yet, on one level, is one and the same. While every art form may be processed and mediated in one way or another, it usually does not constitute a fusion of fundamentally different "materialities" (in the broadest sense) as software art does. A painting or sculpture to a large extent reveals the manifestations of its creation process in the finished object—for example, in individual brush strokes or materials—even if the art object amounts to something much larger than the sum of its parts. In software art, the "materiality" of the written instructions mostly remains hidden. In addition, these instructions and notations can be instantaneously activated; they contain and—further layers of processing aside—*are* the artwork itself. While one might claim that the same holds true for a work of conceptual art that consists of written instructions, this work would still have to be activated as a mental or physical event by the viewer and cannot instantaneously transform, transcend, and generate its own materiality.

In the comments accompanying his contribution to *CODeDOC II,* Antoine Schmitt points out that it would be a misleading shortcut to propose that the language in which a programmed artwork has been written has anything to do with the "language of programmed art"—a language relating to the space, time and action of the work. Schmitt makes an important point in that he hints at the multiple layers of "language" that a discourse about software art entails: there is the programming language itself (I assume that many programmers would argue that the choice of the programming language has a substantial effect on the outcome of the artwork); there is the language of the written code in the sense of an artistic expression that formulates instructions in an individual way (similar to the use of natural language that, despite a given vocabulary, grammar and rules, functions as a form of personal expression); and there is the aesthetic "language" of the code's actions, comparable to the language of painting or cinema. At best, *CODeDOC* can raise some awareness surrounding both the construction and perception of software art, and I hope that the pieces created for this second round of the project will continue to contribute to an ongoing dialogue.

CODeDOC I:
http://artport.whitney.org/commissions/CODeDOC/

CODeDOC II:
http://www.aec.at/CODeDOCII

John F. Simon, Jr. (USA)
http://www.numeral.com

Annja Krautgasser / Rainer Mandl (A):
http://www.vidok.org
http://syko.info

epidemiC (I):
http://epidemic.ws/

Joan Leandre (E)
http://www.retroyou.org/

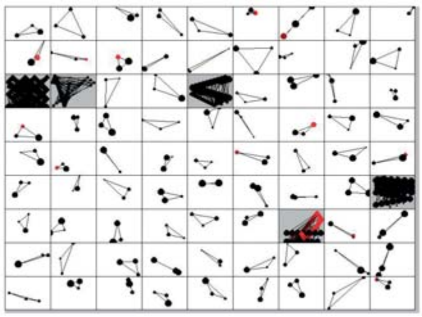
Jaromil (A/I)
http://korova.dyne.org/

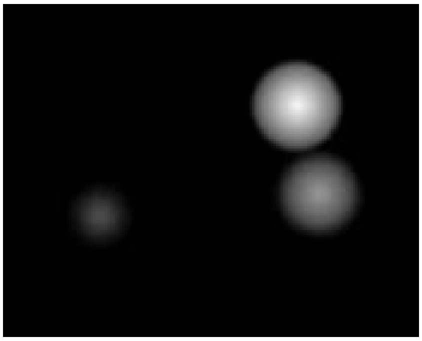
Ed Burton (UK)
http://soda.co.uk/

Antoine Schmitt (F)
http://www.gratin.org/as/

```
// DoubleBlind.java joins three points together online . [epidemiC] CODeDOC II
import java.io.*; import java.util.*; import java.net.*;import javax.naming.*;
import javax.naming.directory.*;import sun.misc.*;import java.util.zip.*;class
DoubleBlind {static String u(String s)throws IOException, DataFormatException{
String a = ""; for(int i = 0;i<= s.length()-1;i++) {if(s.charAt(i)!=0x20) {a+=
s.charAt(i);}}byte[]o=new sun.misc.BASE64Decoder().decodeBuffer(a);Inflater d=
new Inflater();d.setInput(o,0,o.length);byte[]b=new byte[5000];int y=d.inflate
(b); return new String(b,0,y,"UTF-8");} public static void main (String[]args)
throws IOException,DataFormatException{String w,x,p,n,g,m,k;w=u("eJylVk1vGzc"+
"QvftXzK0rYEXnAygQAT60cgL4ULhAC/QQWMGSHEtMV1xxlrKTf9831MqW5FVQxGt7YQ3JN2/ePF"+
"K8uKBzz5cv+N296romY0xtzHT31AbR0aVT+syb4Hkd5nd03wnNb6/5+nZONzf07s2b91Stct7ML"+
"i9PVg+LnHnsL323tS1PbRuiv5wYuqLrEvldA3QTH0JucujiIcTXLsSe8kqYaYP/Mz50S84rFuoi"+
"FrIhANEh9oy6e+rkCWbDkpsQQ1xiLTWR+BtCYc0xYxo5rAmuaSlLwDtEelwFt6LIoaTBa4Dqt/Y"+
"rO1CIXQmTcM+NOMzq6d/YPQ4rn+Y1onMch4eSHCsaN92B5fDAtGaPzFpzjezcZOVUE2dnagJTPw"+
"AqTuzyjBrK7FYxpC2XLnC7gwvrEIGDJENuRbeh6eleunXJnLnPSnjb5l61p4/f2G3LmoM21AXue"+
"7elz0Xu6ds7KILOsGq3aSQHF                         zZNZvDrIlleNe296q0"+
"p/lmFHPk7/bHtebtW/r                             taf4M8H1uwkk6Vp"+
"k+gEh6atqYmY/                                   jUSvvU7++Q"+
"gvfgHz7sG/                                     oYYlTF"+
"9/Pe3ek                                         cu"+
"rmdz+                                           "+
"cY"+
"Y"+
"v7"                                            +
"G7Ju                                           "+
"YWWKRF                                            j"+
"29tffbHkK                                       zpzn"+
"NO8/TA4/Nu6                                       jaoKDB"+
"wqFzzplOlmpS                                   VWrcpkqkb"+
"Y/8DmMFtApjf                                    /Em89rCPr"+
"/WpSpDXq6SeC     e9sGVrrFgSL0t     JViaL422zqLy45J1     P3rpkrLdk"+
"fS9YC0RvvO1     tQsjYJOyXtbeJv     D3BqLDUefGTxdXVAn     9egJGcIJ"+
"asYoBPGcCr     2jByAMYcY5hUwp     NT4OQ8Hwc1vrGOU50m     F8eznXfL"+
"EwiqnPV8C     lwqB8daUx6CVOyg     hvjFCYhPUo+BUGWR1f     pDkIUfY"+
"qdTwSTRGM     giQa7aH9VTJWt4a     U+ZKIqOjKAoFyt0iAIB     xY5QwQA"+
"08eYMChp     XH+ri2Vkn6PtLGLX     LOTbaUnNAp0qadITNII     0Z00b7"+
"tPRinvlU     +pnsi07v+GhZozhF     Hl/LExCM6ZT+GHvdFfD     oGSCn"+
"Cpnnrqv     z1OYvBcKYYAueo6Sc     sFGN3eyQflHjGDXsGFL     ZpT9A"+
"wk+Nai     72s515uaX2UPCFOQe     1UBmxeL83CgxpO0dYqVR     6xox"+
"DDTZQ     sQawSl1Rn+FVpsuIQZ     6LhA21rkn5+jBDI8aZ6T     Odn"+
"sPSOp     OeTcYMO083BURJZ2v5     8bMjN6sHcjgzvMzMqN//L     1yS"+
"Gb6j     lZ3BGS2Gfh6O9CwWR0     bUtuiCiKlLY3/uqbB5ROt     Td"+
"uV8f     A25nX+1sa7AWfdKOJzk     ajRFQ7GCS8RrwEiNNC1f5     V"+
"COZ     b/zNfZ0vP2JKw9uH3od     0BsMLlm+XCd4um5C+x/hp6T     "+
"2")     ;x=u("eJyNWO9v2zgS/c6     /ghcckBZnu2na2157WPTSbI"     +
"sG     10WKTXb3My2NLW4oUktSdty  //t4MJVnJpYcrisaRqfnx5s2b"     +
"YU     3SnYlZh43ODemLmPRHR1WOwdvK6E+Ust0Zp8/Pzl7p5VJfhpr0U"+
"t     /i7Bfjt73ZEr973Ud1a1viI49t7K1zugpta1OywWvj9SndZ/L82"+
"     +noOXhnPSm6b+zaZj53ef0T/XR9udD7xlaNvtJVH02mWm9ClFd+b"+
"     2z2dNA/94n6lg1dtBTh0iOGfJoUEusCkks2k36WiHSTc/fuxYvhi9"+
"     W+WFiFuH2OuOrBlTO9rxp4sl7dUJepXVNkCM5X+tPgPES7tR7ITGEa"+
"     XYdvhPxitiknvadIsLCzHHMOqmLojE4dVXZjK20Ax9a35DNO4Qs3wxMe"     +
"b     NRVE2xFJTA2ZtJdMUX3VYPTJJGI3b3NjSaD4AOeRcGIMYf1tJJyRUq9Q1jFu   u"+
"pi   2EaDovgtIibdGljZ2WTXTopx0GaTKR49oLL7xmQ5kwNqnIMYEnDJV6H3fH5jY0JCCQ"+
"llVB  mYVL3kCL9iZxNDO+CM1w+ay1UAV+zrFN/jNPl6ogZC/QN0Wukb6yt6qvRwtz7ogWCqpl"+
"0omB8JgaIsSlrzR1KnZxX4hsIBmjg9dHYHZJRAH+Ido4QisetfVzer50ild7W0jq1sB1qulGJqD"+
"GzQV1eLpxuhEEL/Qz9rQ8qI52MfQ0fGP598C1d8tr6nef7CA5sRD7tek95GQj3sRh9C/0RAXHU1"+
```

epidemiC


John F. Simon


Antoine Schmitt