# *CODeDOC II*

**Christiane Paul**

Im Rahmen der CODE-Ausstellung, die während der diesjährigen Ars Electronica stattfindet, wurde ich eingeladen, eine Fortsetzung der Online-Ausstellung *CODeDOC I* zu kuratieren, die ich ursprünglich für das *artport* des Whitney Museum of American Art organisiert hatte, eine Website, die als Portal für Netzkunst entworfen wurde. *CODeDOC* startete im September 2002 und sollte die Beziehungen zwischen den der Softwarekunst zugrunde liegenden Codes und ihren Ergebnissen untersuchen. Etwa ein Dutzend Softwarekünstler wurde eingeladen, in einer Programmiersprache ihrer Wahl (Java, C, Visual Basic, Lingo, Perl) den Code für eine bestimmte Aufgabe zu schreiben, nämlich „drei Punkte im Raum zu verbinden und zu bewegen". Darüber hinaus sollten die Codes untereinander ausgetauscht und kommentiert werden.

Das Präsentationskonzept von *CODeDOC* unterscheidet sich bewusst von den Formen, in denen Betrachter Softwarekunst im Allgemeinen erfahren. Sie wird dem Publikum meist als ausgeführter Code – als Resultat schriftlicher Anweisungen – präsentiert. In *CODeDOC* ist die Rezeption dem künstlerischen Schaffensprozess näher: Das Publikum sieht zunächst eine Seite mit dem geschriebenen Code, von der aus er sich auch ausführen lässt. Da die Aufgabenstellung eine beträchtliche Beschränkung in Format und Dateigröße bedeutete, sind die Projektbeiträge nicht unbedingt als zur Gänze ausgearbeitete Werke zu betrachten. Sie sind vielmehr kleinen Studien und Skizzen vergleichbar, die den künstlerischen Ansatz wiedergeben. Viele international herausragende Vertreter der Softwarekunst konnten am ersten *CODeDOC* nicht teilnehmen, da das Whitney Museum – seinem Auftrag gemäß – amerikanischen Künstlern (Bürgern und Künstlern, die in den USA leben und arbeiten) vorbehalten ist. *CODeDOC II* ist eine willkommene Möglichkeit, diese Lücke zu füllen und den Rahmen des Projekts zu erweitern. Die acht Künstler / Teams, die für die zweite Folge eingeladen wurden, kommen zum Großteil nicht aus den USA: Ed Burton, epidemiC, Graham Harwood, Jaromil, Annja Krautgasser & Rainer Mandl, Joan Leandre, Antoine Schmitt und John F. Simon jr. Einige weitere Künstler, die sich für dieses Projekt angeboten hätten, konnten nicht berücksichtigt werden, weil sie bereits in anderen Bereichen der Ars Electronica oder in der Ausstellung vertreten waren. Mein besonderer Dank geht an Andreas Broeckmann für seine Anregungen und Vorschläge bei der Auswahl der Künstler.

*CODeDOC* war von Anfang an eher als prozessorientiertes Experiment denn als Ausstellung, die bestimmte Aussagen oder Standpunkte zum Inhalt hat, konzipiert. Idealerweise wollte ich Fragen zur Softwarekunst als künstlerische Praxis aufwerfen, wobei weder das Ergebnis noch die Rezeption des Projekts für mich absehbar waren. Eine Intention des Projekts war zweifelsohne, den Code als „mysteriöse", unsichtbare treibende Kraft zu entmystifizieren und dem Betrachter näher zu bringen. Fragen, die anzuschneiden oder zu klären, mir wichtig schien, waren u. a. die folgenden: Beschreibt der Begriff „Softwarekunst" an sich eine bestimmte Form der Ästhetik? Manifestieren sich „Handschrift", „Stimme" und Ästhetik eines Künstlers im geschriebenen Code und den ausgeführten Ergebnissen? Verbessert das Verständnis des Quellcodes die Wahrnehmung der Arbeit? Handelt es sich dabei um eine wirkliche Erweiterung oder lediglich um eine Akzentuierung „technischer Einzelheiten", die unnötig und befremdlich ist und die Arbeit verschleiert? Wie könnte man die Beziehung zwischen

dem Back-End des Codes und seinen Ergebnissen definieren?
Der Versuch, detaillierte Antworten auf all diese Fragen zu geben, würde den Rahmen dieser Einleitung sprengen, weshalb ich mich auf einige allgemeine Anmerkungen beschränken und es den *CODeDOC II*-Projekten sowie den sich daraus ergebenden Diskussionen überlassen möchte, weitere Perspektiven zu diesen Themen aufzuzeigen.
Untersucht man die Gesamtheit der Arbeiten, die jeder *CODeDOC II*-Teilnehmer im Lauf der Jahre schuf, wird deutlich, dass das Etikett „Softwarekunst" eher der kleinste gemeinsame Nenner einer formalen Beschreibung ihrer künstlerischen Praxis als ein Terminus für eine bestimmte Ästhetik ist. Die Arbeiten der Künstler decken ein breites Spektrum sehr individueller Ansätze ab. Die Arbeiten von epidemiC etwa – *AntiMafia,* ein Windows-basiertes Programm zur Koordination assoziativer Aktionen sowie der berüchtigte *biennale.py*-Virus (eine Kooperation mit 0100101110101101.ORG), der für die 49. Biennale von Venedig entstand – setzen eher auf Aktivismus und das Konzept von Software als einer kulturellen Produktion. Ed Burtons *Sodaplay* und *Sodaconstructor,* die mittlerweile Kultstatus haben, untersuchen die konzeptuellen Möglichkeiten „handgefertigter" virtueller Roboter sowie von Massen und ihrer kinetischen Energie. Graham Harwoods Arbeiten reichen von „reiner" Perl-Poesie bis hin zu Softwareproduktion und narrativen Projekten wie etwa der CD-ROM *Rehearsal of Memory* (für die er mit einer Gruppe aus dem Ashworth Hochsicherheitsspital ein interaktives Programm erstellte, bei dem die Haut der Projektteilnehmer gescannt wurde, um physische Spuren ihres Lebens als Geisteskranke festzuhalten), oder dem Webprojekt *Uncomfortable Proximity,* einer Auftragsarbeit für das Tate Museum, bei der das Layout, die Logos und das Design der Tate-Website reproduziert wurden, um die Geschichte des britischen Kunstbetriebs aus einem „anderen Blickwinkel" zu erzählen. Verglichen mit den vorgenannten Beispielen sind Antoine Schmitts Arbeiten eher visuell orientierte Studien über das „Verhalten" von Formen in Zeit und Raum.
Obwohl man meinen könnte, dass sich der Ansatz eines Künstlers (und vielleicht sogar seine „Persönlichkeit") gleichermaßen im geschriebenen Code und dessen Ergebnissen manifestiert, ist der Code natürlich für andere Programmierer interessanter als für ein allgemeines Publikum, das vielleicht nur eine vage Vorstellung von dessen „Mechanismen" bekommt. Ob der Code zum Verständnis der Arbeit beiträgt, variiert auch stark von Fall zu Fall. Es drängt sich der Gedanke auf, dass die Bedeutung, die die Künstler selbst dem Code zuweisen, auch vom Charakter der jeweiligen Arbeit abhängt: Für Künstler, deren Werk sich auf den „unverfälschten" Code konzentriert (wie etwa viele Arbeiten von Graham Harwood), mag der „schriftliche Teil" des Projekts wichtiger sein als für Künstler, deren Werke visuelle Formen, Raum und Aktion erforschen (wie viele Projekte von Antoine Schmitt). Das Format der Präsentation von *CODeDOC* hat die Künstler anscheinend (unbeabsichtigterweise) zu Editierungen veranlasst: In ihren Kommentaren haben sowohl Antoine Schmitt *(CODeDOC I)* als auch Camille Utterback *(CODeDOC I)* zugegeben, dass sie sich bemüßigt fühlten, ihren Code zu „säubern", bevor sie ihn dem Publikum präsentierten („Ich gehöre zu den Leuten, die ihr Badezimmer reinigen, wenn Freunde zu Besuch kommen", wie Camille es formulierte). Eine der inhärenten Gefahren und unbeabsichtigten Effekte von *CODeDOC* wäre die falsche Annahme, dass die Qualität von Softwarekunst nach der Virtuosität und technischen Fertigkeit im Programmieren beurteilt wird (also nach den von Donald Knuth skizzierten Kriterien wie Richtigkeit, Wartbarkeit, Übersichtlichkeit und Lesbarkeit). Eine der Schönheiten von Kunst besteht, ungeachtet ihrer Ausdrucksform oder des verwendeten Materials, darin, dass ihr Erfolg auf zahlreichen Faktoren beruht, die nicht objektiv zu definieren sind. Ein Betrachter kann das Werk Leonardo da Vincis oder Picassos sicher aufgrund ihrer herausragenden Virtuosität und ihres handwerklichen Könnens genießen (wenngleich sie viel mehr zu bieten haben), diese Maßstäbe versagen aber bei Duchamps Urinal oder Beuys' Skulpturen aus Filz und Fett. Wie jede andere Kunstform kann und sollte Softwarekunst nicht auf technische Kriterien reduziert

werden, und der Code ist mehr als nur das Getriebe, das die Maschine in Gang setzt.
Offensichtlich unterscheidet sich Softwarekunst als künstlerisches Medium und in seiner Praxis von anderen Kunstformen wie Malerei, Bildhauerei oder Film/Video. Im Gegensatz zu anderen visuellen Kunstformen schreiben Softwarekünstler ausführbare verbale Anweisungen für ihre Arbeiten, die alles von visuellen Lösungen bis hin zu abstrakteren Kommunikationsprozessen produzieren (wenngleich die Ausführung eines Codes nach wie vor verschiedene Interpretations- und Kompilationsschritte erfordert und der Code selbst größtenteils eine logische Notationsform darstellt). Es besteht eine sonderbare Beziehung zwischen dem größtenteils verborgenen Back-End des Codes – der eine Konvergenz von Sprache und Mathematik darstellt – und dem multisensorischen „Display", das er erzeugen kann: eine „Identität" im Sinne einer Übereinstimmung verschiedener Instanzen (Code-Ergebnisse), die jeweils eine ganz unterschiedliche Form annehmen, auf einer Ebene aber doch ein und dasselbe sind. Während alle Kunstformen auf die eine oder andere Weise bearbeitet und vermittelt werden können, stellen sie im Unterschied zur Softwarekunst im Allgemeinen keine Verschmelzung von im Wesentlichen unterschiedlichen „Materialitäten" (im weitesten Sinne) dar. Bei einem Gemälde oder einer Skulptur manifestiert sich der schöpferische Prozess im fertigen Objekt – beispielsweise in den einzelnen Pinselstrichen oder im Material –, auch wenn das Kunstobjekt mehr ist als die Summe seiner Teile. In der Softwarekunst bleibt die „Materialität" der geschriebenen Anweisungen größtenteils verborgen. Darüber hinaus können diese Anweisungen und Notationen unverzüglich aktiviert werden, sie enthalten und sind – abgesehen von weiteren Bearbeitungsschritten – bereits das Kunstwerk. Zwar könnte man einwenden, dass das auch für aus geschriebenen Anweisungen bestehende Konzeptkunst gilt, doch müsste ein solches Werk noch in einem geistigen oder körperlichen Akt durch den Betrachter aktiviert werden und kann nicht sofort seine eigene Materialität transformieren, transzendieren und generieren.
In den begleitenden Kommentaren zu seinem Beitrag für *CODeDOC II* weist Antoine Schmitt darauf hin, dass es eine irreführende Verkürzung wäre zu glauben, dass die Sprache, in der ein programmiertes Kunstwerk geschrieben wurde, etwas mit der „Sprache programmierter Kunst" zu tun hätte – einer Sprache, die sich auf den Raum, die Zeit und die Wirkung der Arbeit bezieht. Schmitt verweist auf die multiplen Schichten der „Sprache", die ein Diskurs über Softwarekunst aufzeigt: Da ist einmal die Programmiersprache selbst (ich gehe davon aus, dass viele Programmierer die Auffassung vertreten, dass die Wahl der Programmiersprache eine wesentliche Auswirkung auf das Ergebnis des Kunstwerks hat), dann die Sprache des geschriebenen Codes im Sinn eines künstlerischen Ausdrucks, der Anweisungen auf individuelle Weise formuliert (ähnlich der natürlichen Sprache, die trotz der Vorgabe von Vokabular, Grammatik und Regeln auch eine persönliche Ausdrucksweise zulässt), und schließlich noch die ästhetische „Sprache" der Wirkung des Codes, die der Sprache der Malerei oder des Films vergleichbar ist. Ich hoffe, dass *CODeDOC* das neue Einblicke in die Produktion und Rezeption von Softwarekunst gewährt, und dass die Arbeiten, die für diese zweite Runde des Projekts entstanden, zur Weiterführung des Dialogs beitragen.

Aus dem Amerikanischen von Martina Bauer