4. Literarische Computerspiele und Program Code Poetry

4. 1. Literarische Computerspiele

Jim Andrews' "Arteroids 2.03. A literary computer game for the Web" ist eine verbale Adaptation des klassischen Videospiels "Asteroids". In "Arteroids" kämpft man gegen Wörter, die aus dem off in den Bildschirm driften und das Wort 'poetry' bedrohen, da ein Kontakt mit den Invasoren die Explosion von 'poetry' verursacht. Das Wort 'poetry' kann gesteuert werden, insbesondere kann man den Wörtern ausweichen. 'Poetry' kann aber auch als Kanone eingesetzt werden, die die verbalen Invasoren abschießt. Die getroffenen Wörter explodieren ebenfalls, die einzelnen Lettern bleiben im Raum hängen. Laut Jim Andrews handelt es sich bei seiner Erfindung um "kind of an 'end of language' piece". Das alte Klischee, nach dem der Tod der Feind der Poesie ist, wird hier nicht mehr dargestellt, sondern performiert, und wir werden als *user* in die Schlacht hineingezogen. Als Invasoren treten unter anderem die Wörter 'death', 'fear', 'insecurity', 'nothing' und 'poetry' in Erscheinung. Mit diesen Wörtern kann man Sätze bilden wie "The battle of poetry against itself and dullness". Aber die Aufmerksamkeit der *user* wird unweigerlich vom Lesen ab- und zum Kämpfen hingelenkt. Und das ist gut so, wenn man überleben will, Lesen ist hier gewissermaßen lebensgefährlich.
Um die Immersion zu steigern, kann die Sprache geändert werden, und zwar von Englisch auf Portugiesisch, man kann die invadierenden Wörter vorgeben und auch der 'poetry'-Kanone einen Namen geben, zum Beispiel den eigenen. Auch die Dichte und Geschwindigkeit der Invasoren kann gesteigert werden - "low density is readable", "high density menaces id entity in hostile word hordes" bemerkt Andrews dazu -, ganz allgemein kann man zwischen dem leistungsorientierten 'game mode' und dem literarisch-beschaulichen 'play mode' wählen. Im 'play mode' kann man eigene Wörter vorgeben, die entstehenden Texte editieren, in einem Editor mit dem schönen Namen "Word for Weirdos" speichern und per e-mail verschicken. Zusätzlich hat Andrews 21 Audio-Dateien mit verschiedenen 'Todesschreien' eingebaut: die getroffenen Wörter schreien mit männlicher oder weiblicher Stimme, mit junger oder alter, menschlicher oder 'semi-human' Stimme. Die *sounds* sollen das Spiel nach dem Willen seines Schöpfers "ultra human or hyperhuman" machen.
Arteroids
"Arteroids" ist ein ironisches Remake von Computerkampfspielen, den viel diskutierten Ego-Shooters, deren Rhetorik verfremdet wird. Als Spieler/Leser sollte man seine Identität, seinen Avatar, in der Spielwelt involvieren. Alles soll sehr 'ursprünglich' sein, Andrews spricht in diesem Zusammenhang von 'primality' in der Kunst und behauptet, dass "guns are primally dramatic". In der Installation wird die Verbindung von (leistungsorientiertem bzw. spielerischem) Spiel und Kunst erprobt. Auf Wörter zu schießen, mag auf den ersten Blick lächerlich erscheinen, aber es wirft die Frage auf, was Spielen von Lesen unterscheidet und ob das Moment des Abenteuerlichen nicht immer zum Lesen gehört.

Vergleichbar, aber vom Design her viel bescheidener ist Johannes Auers "Kill the Poem". Man schießt auf einen permutativ generierten Text - "keine faxen mit tango ist ernst kein tango ist ernst mit faxen keine faxen ist tango mit ernst mit tango ist ernst ohne faxen …",

löscht zuerst das Wort ‘faxen’, dann ‘ohne’, ‘mit’ und so weiter, bis der gesamte Text ‘gekillt’ ist. In einem Kommentar hat Reinhard Döhl, Auers Stuttgarter Kollege, das Gedicht als Demonstration der Dekonstruktion eines Kunstwerks interpretiert (hat er etwa Destruktion gemeint?). Andererseits wird auch die Unzerstörbarkeit des Gedichts vorgeführt, denn der Text kann natürlich sofort wieder ‘nachgeladen’ werden (wie auch der vom Wurm gefressene Apfel in einer Schleife sofort wiederkehrt). Und Döhl verweist auf die Vorläufer dieser Idee, zum Beispiel Hans Arps Vorschlag, Laokoon ein Klistier zu verpassen, damit er nach Jahrtausenden Kampf gegen die Schlange endlich die Toilette aufsuchen könne und Marcel Duchamps bärtige Mona Lisa, die1965 ‘rasiert’ wurde.
Kill the Poem

Der Tod (der Poesie) scheint ein anziehendes Sujet für digitale Poeten zu sein. Manfred Arens verwendet eine Passage aus der Einleitung zu Boccaccios *Decamerone* für eine Installation kinetischer Poesie mit spielerischen Komponenten. “Die Phasen der Pest” verwendet den Text

e fu questa pestilenza
di maggio forza per
ciò che essa dagl’infermi
di quella per lo comunicare
insieme s’avventave a’ sani

der in der Übersetzung von A. W. Schlegel lautet:

Die Auswirkung dieser Seuche
war verheerend, da sie schon
durch Umgang mit einem Kranken
auf die Gesunden übersprang.

Die “Phasen der Pest” ist ein gutes Beispiel eines performativen Textes: Die schwarzen Buchstaben färben sich rot, was bedeutet, dass sie infiziert sind, manche werden grau, das heißt, sie sind gestorben. Der Text simuliert damit den Fortschritt der Seuche. Arens beschreibt “Die Phasen der Pest” als Feld rechteckiger Zellen, die ihren Zustand unter dem Einfluss benachbarter Zellen ändern. Die Buchstaben werden nach dem Zufallsprinzip von links, rechts, oben oder unten infiziert (die Pest breitet sich also ‘nicht-linear’ aus), sie gehen an der Pest zugrunde oder sie werden geheilt.
Als *user* kann man Schicksal spielen und die Infektions- und Todesrate und auch den Grad an Zufälligkeit bestimmen. Ferner können die Buchstaben auch immunisiert werden. Bei einer Infektionsrate von 0% wird nur der erste Buchstabe (das e) angesteckt, die Pest wird also sofort gestoppt, bei einer Todesrate von 100% ‘sterben’ alle Buchstaben und der Bildschirm leert sich, nur die Anführungszeichen und Kommata bleiben übrig.
Die Phasen der Pest

4. 2. Computerspiele als Literatur?

Abgesehen von solchen Installationen, in denen mit Texten spielerisch umgegangen werden kann, sind auch regelrechte Computerspiele als Texte bzw. Erzählungen (narrative Medien) betrachtet worden. Grundsätzlich können Textspiele (wie der Klassiker *Zork*) und grafische Spiele unterschieden werden. Textspiele werden heute nicht mehr produziert. Grafische Spiele werden in 2D- und 3D-Spiele unterteilt, ferner in 1st-person- und 3rd-person-Spiele. Der Unterschied besteht darin, dass man sich als Spieler selbst sehen kann oder nicht. Nach dem Aufbau der Spiele können Action-, Adventure-, Strategie-, Rollenspiele und Simulationen unterschieden werden. Die stärkste Narrativität besitzen Adventure-Spiele (Bsp. *Zork*), sie können am ehesten mit Literatur verglichen werden. Man führt seine Figur durch die Geschichte, löst diverse Rätsel oder Puzzles, sucht Dinge und wendet sie an und arbeitet sich von Level zu Level hoch. Bei Rollenspielen (Bsp. *Gothic*) entwickelt man den Charakter seiner Figur im Lauf des Spiels. Strategiespiele basieren vollkommen auf den Entscheidungen der *user*, sie sind sehr offen und entsprechend fern von Narrativität und Literatur. Bei Simulationen ist das narrative Moment meist gering, eine Ausnahme bildet hingegen Sims. Bei Actionspielen schließlich (Bsp. *Quake*) ist der Grad an Interaktivität hoch, die Narrativität bestenfalls mittelhoch.
In der Computer-Spielforschung unterscheidet man zwei Lager: die Narratologen und die Ludologen. Die Naratologen betrachten die Spiele - ähnlich wie Literatur und Film - als *storytelling medium*, die Ludologen kehren den Spielcharakter in den Vordergrund. Zu den Narratologen zählt Janet H. Murray mit ihrem Buch *Hamlet on the Holodeck. The Future of Narrative in Cyberspace* (1997). Ihrer Ansicht nach erzählen alle Spiele eine Geschichte - und sei es die eines simplen Wettbewerbs um den Sieg in einem Sportspiel. Computerspiele ordnet sie zusammen mit Internetcommunities, interaktiven TV-Sendungen und anderem unter der Rubrik *cyberdrama* ein. Digitale Welten sind procedural, participatory, spatial und encyclopedic, Eigenschaften,die man auch als interaktiv und immersiv zusammenfassen könnte. Sie bilden die logische Fortführung der alten Medien, entwickeln deren Möglichkeiten fort und passen sie an die Gegebenheiten des 21. Jahrhunderts an.

> In a postmodern world, however, everyday experience has come to seem increasingly gamelike, and we are aware of the constructed nature of all our narratives. […] Therefore the union of game and story is a vibrant space, open to exploration by high and low culture, and in sustained and incidental engagements by all of us as we negotiate the shifting social arrangements of the global community and the shifting scientific understandings of our inner landscape. […] The digital medium is the appropriate locus for enacting and exploring the contests and puzzles of the new global community and the postmodern inner life.

Aus der ludologischen Sicht Espen J. Aarseths gibt es interaktive Erzählungen (cybertexts), an deren Konstitution die *user* beteiligt sind, unabhängig von den Medien, in denen sie auftreten (man erinnere sich z. B. an die Hypertexte auf Papier). Das Konzept der Leseraktivierung als entscheidendes Merkmal der Abgrenzung erinnert stark an Ecos offenes Kunstwerk.

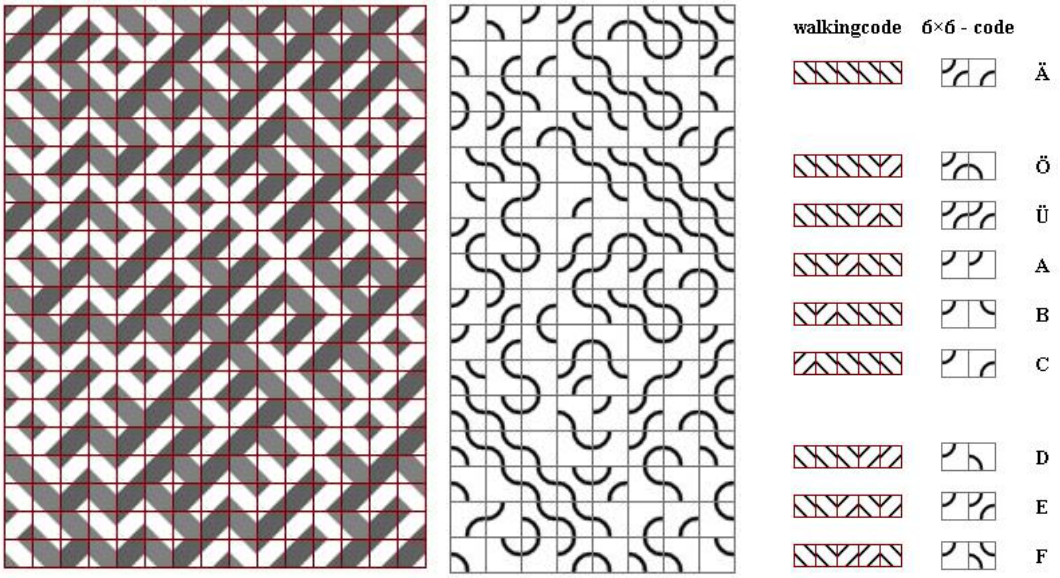
> The concept of cybertext focuses on the mechanical organization of the text, by positioning the intricacies of the medium as an integral part of the literary exchange. […] The performance of the reader takes place all in his head, while the user of cybertext also performs in an extranoemic sense. During the cybertextual process, the user will have effectuated a semiotic sequence, and this selective movement is a work of physical construction that the various concepts of "reading" do not account for.

Computerspiele als Literatur zu betrachten, bezeichnet er als akademischen Kolonialismus und betont ihre Eigenständigkeit. Statt von *story* spricht er lieber von *intrigue*, die Handlungen des Spielers werden wie in einem Logbuch aufgezeichnet und ergeben den *ergodic discourse*. Hinter allen Computerspielen stehe das Prinzip der Simulation.
Für die Position der Ludologen spricht, dass Spiele sich nur schwer in andere erzählende Medien übertragen lassen – und umgekehrt. Ein Spiel wie *Star Wars* hat mit dem Film nur den Namen gemein.
Fraglos gibt es aber so etwas wie Erzählmuster in Spielen, auch wenn sie manchmal nur ganz basal sind. In Kontakt mit der *story* kommt man meist auf der Verpackung (vergleichbar den *blurbs* oder Klappentexten der Literatur), dann im Handbuch. Besonders literarisch sind natürlich Textspiele, von denen manche ausschließlich mit Worten arbeiten. Aber auch in anderen Spielen können Tagebuchaufzeichnungen, Briefe, Manuskriptfunde u. ä. eingeschaltet werden. Auch Dialoge, vor allem solche mit Nebenfiguren, dienen häufig der Information über *story*. Analoges gilt auch für grafische Elemente: auch sie vermitteln Informationen über die Geschichte, ähnlich wie Buchillustrationen. Man begegnet ihnen auf der Verpackung oder sie sind in einem Vorspann, einer Art Trailer, enthalten. Selbstverständlich treten gelegentlich auch Erläuterungen über Stimmen aus dem off auf.
Für die Position der Narratologen spricht, dass sich in Computerspielen ähnliche Perspektiven finden wie in literarischen Erzählungen.
- Nullfokalisierung, das heißt der Erzähler weiß mehr als die Figuren wissen können. Da Spiele keinen Erzähler kennen, entsteht Wissen über die Figuren z. B. durch die Bedienungsmöglichkeiten und die Struktur des Spiels. Nullfokalisierung liegt häufig bei Strategiespielen und Simulationen vor. Die Spieler werden hier fast selbst zu Erzählern.
- In Sims, das als Simulation des Lebens von durch die *user* kreierten Personen in einer ebenfalls frei gestaltbaren Umwelt handelt, 'erzählen' und 'lesen' die Figuren gleichzeitig, ist der Spieler so etwas wie der 'Gott' dieses künstlichen Universums; allerdings kann es passieren, dass die Figuren plötzlich etwas völlig Unerwartetes tun. Dann kippt die Perspektive zu einer externen Fokalisierung, in der die Figuren mehr 'wissen' als der Erzähler.
- Der häufigste Fall ist sicher die interne Fokalisierung, bei der der Informationsstand von Erzähler und Figuren gleich ist. In Adventurespielen ist es zum Beispiel die Regel, dass man als *user* – zumindest beim ersten Mal - keinen Überblick über das ganze Spiel besitzt und nicht erwartet, dass die Figuren eigenmächtig handeln.
Eine personale Erzählsituation tritt auf, wenn man in einem Adventurespiel nur über die Sichtweise einer Figur verfügt, etwa in Form einer 3rd-person-Perspektive. Action-Spiele mit einer 1st-person-Perspektive gehören klarerweise zur Ich-Erzählsituation, man 'ist' als Spieler

zugleich Hauptperson. Die auktoriale Erzählsituation, in der ein Erzähler aus einer Außenperspektive berichtet, trifft wiederum auf Strategiespiele und Simulationen zu.

4. 3. Program Code poetry

Auf seinen Internetseiten mit dem Titel “untexte” weist Manfred Arens darauf hin, dass die natürlichen Sprachen nur eine Möglichkeit unter unendlich vielen Möglichkeiten der Kodierung sind. Er überträgt Gedichte in verschiedene Codes wie musikalische Notenschrift, farbige Stäbchen, Rechtecke, Bodenfliesen und andere graphische Muster.



Die obigen Bilder sind verschlüsselte Gedichte. Wenn man den Code kennt, der ein 2^5- oder walkingcode ist, kann man das linke Bild als “Wer will erfaren der Welt Wesen, der thuo disen Reimen lesen” entschlüsseln. Das mittlere Muster repräsentiert den Satz “Der untext verhält sich zur leeren Seite, wie der Text zum Untext”. Es ist in einem Code, den der französische Mönch Sébastien Truchet im 18. Jahrhundert für Fliesen entwickelte nachempfunden.
LABOR INTUS
Die von Arens verwendeten Codes könnten ohne weiteres zur Steuerung von Computern verwendet werden. Sie weisen auf die doppelte Codierung von digitalen Texten hin. Unter der Oberfläche der natürlichen Sprache verbirgt sich eine Kunstsprache, die Programmiersprache, die ihrerseits für den Rechner wieder in binären Code umgewandelt wird. Es ist daher wenig überraschend, wenn Cyber-Autoren versuchen, Programmiersprachen poetische Effekte zu entlocken oder zumindest mit Elementen von Programmiersprachen zu spielen. Man spricht in diesem Fall von *Program Code Poetry* oder *Code Work*. Man hat diese Dichtungsgattung definiert als “literature which uses, addresses and incorporates code [...] as a special type of language in itself, or as an intrinsic part of the new surface language or ‘interface text’, as I call it, of writing in networked and programmable media.” Auch diese Experimente weisen auf den Unterschied zwischen Oberflächentext und der Tiefenstruktur, die ihn generiert, hin, zwischen menschlicher Lektüre und Maschinentext. Die Programmiersprache, in der der

Source Code verfasst ist, nimmt eine vermittelnde Rolle zwischen der für uns lesbaren Oberfläche und der Maschine ein. Jaromil, ein Netzkünstler, bemerkt dazu:

> Source code means a formulation of 'instructions' expressed in a language understandable to a computer and linked in accordance with logical and conditional patterns which, once interpreted and executed, gives rise to a result. [...] Every language is defined by a grammar which, in turn, is interpreted by a compiler who 'metabolizes' its semantic content (instructions) and so produces a 'bytecode' which the computer can execute.

Und Lawrence Lessig erläutert die Schwierigkeitsgrade der beiden Schichten von Code unter der Oberfläche:

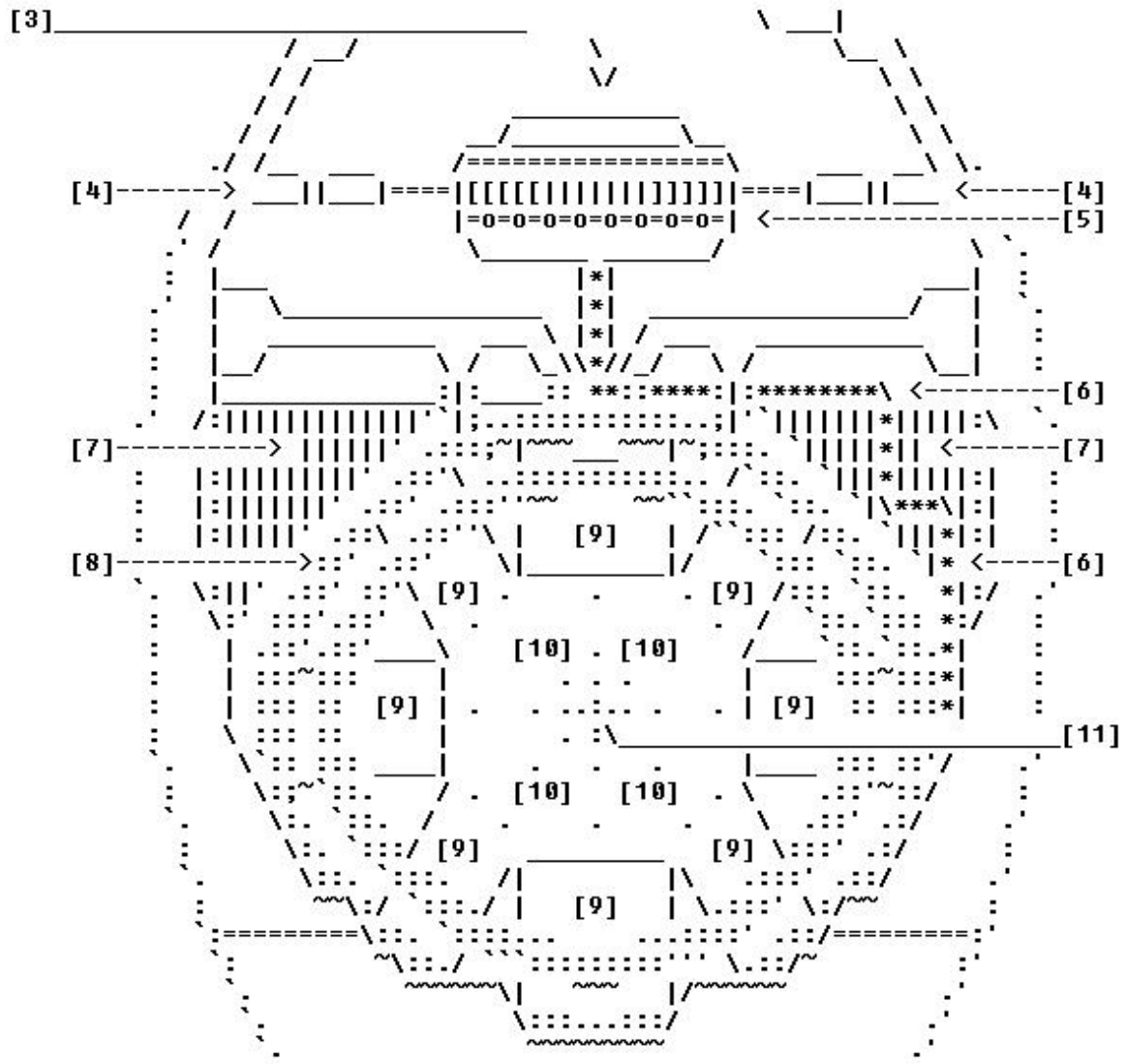
> Source code is the code that programmers write. It is close to a natural language, but not quite a natural language. A program is written in source code, but to be run it must be converted into a language the machine can read. Some source code is converted on the fly [...]. But most source code - or the most powerful source code - is 'compiled' before it is run. The compiler converts the source code into either assembly code (which mavens can read) or object code (which only geniuses and machines can read). Object code is machine-readable. It is an undifferentiated string of 0s and 1s that instructs the machine about the tasks it is to perform. Programmers do not directly write object code, even if some are able to decipher it; programmers write source code. Object code speaks to the computer; source code speaks to humans and to computers (compilers); assembly code speaks to mavens and computers. (103)

Der *Source Code* oder Quellcode repräsentiert sozusagen die unsichtbare und für den Laien geheime Welt der Maschine, in der Bedeutung durch strikt logische Prozesse generiert wird. Wenn *Source Code*, ein Programm, 'läuft', bewirkt es etwas, und zwar im ganz materiellen Sinn. Daher hat man ihn mit illokutionären Sprechakten verglichen, die zur gleichen Zeit etwas aussagen und bewirken bzw. performieren. Gerade diese Performativität der Sprache wird betont, wenn der *Source Code* für künstlerische Zwecke an die Oberfläche geholt wird.

Jodi ist ein Akronym von Joan Heemskerk und Dirk Paesmans. Dieses holländisch-belgische Künstlerduo betreibt verschiedene Netzkunstprojekte. Hier interessieren uns ihre Installationen, in denen sie den ASCII-Code zur Herstellung von Texten oder Bildern in der Tradition der Konkreten Poesie verwenden. ASCII (American Standard Code for Information Interchange) umfasst die 128 Buchstaben, Zahlen, Interpunktions- und Zusatzzeichen einer amerikanischen Schreibmaschinentastatur. Dieser Zeichensatz bildet nach wie vor den Basiscode für den e-mail-Verkehr und die Programmiersprachen. Viele von Jodis Arbeiten sind a-semantische Zeichenketten, die aussehen wie der Bildschirm eines infolge Systemfehlers abgestürzten Computers. Jodi durchbrechen die Oberfläche des Informationssystems und simulieren ein Code-Chaos. Wenn man den *Source code* betrachtet, verwandelt sich das scheinbare Chaos in Ordnung oder zumindest in ein Muster. Die normalen Verhältnisse - lesbarer Oberflächentext, kryptischer Quellcode - sind hier umgekehrt. Die Zeichen sind dieselben geblieben, aber ihre Anordnung, die durch den

Webbrowser zerstört wurde, ist jetzt eine andere und die ergibt Sinn oder ein Bild wie im folgenden Beispiel. Zuerst wird im Webbrowser ein “Zeichensalat” sichtbar.

Erst der Quellcode ergibt in gewisser Hinsicht “Sinn” oder jedenfalls ein Muster, man könnte auch sagen: ein Kunstwerk.



ASCII-Kunst geht zurück auf Versuche, mit der Tastatur zu zeichnen, und sei es so einfache Dinge wie :-). Aufwendiger sind Projekte wie *Deep ASCII*, ein von der Gruppe ASCII hergestellter Text-Film, der das Vorbild, den Pornofilm *Deep Throat*, digitalisiert und zu einem bewegten Zeichenraster umwandelt. Es ist kaum etwas zu erkennen, außer vagen Umrissen und beweglichen Formen. Entfernt erinnert das Ergebnis an Bilder von verschlüsselten Pay-TV-Sendern. Auch hier geht es um Fragen der Verschlüsselung und Dekodierung.
Deep ASCII

Im digitalen Medium sind alle Gegenstände (auch Bilder, Musik …) unter der Oberfläche textuell. Kodierung und Darstellung treten weit auseinander. Die normalerweise verdrängten Quellcodes werden an die Oberfläche geholt, einschließlich der Fehlermeldungen und Störfaktoren, deren subversives Potential entsprechend betont wird.
Der Kanadier Ted Warnell etwa verfremdet in seinem Text DAT CiDE NULi: eine Raumschiff-Enterprise-Dialogzeile (“you will be assimilated - resistance is futile” zu “resistance is fertile”).

```
/*      i  .     /2    DAT CiDE NULiXiXioXXXXXX.960509[ resistance
]                                         XXXXXXXXXXXXXXXX
]                                         XXXXXXXXXXXXXXXX
]                                         XXXXXXXXXXXXXXXX
 =CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=
=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=
=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD[
   =95o=95=95 =95=95=95=95=95=95=95=95=95=95 =95=95=95=95=95=95 ][    =95o=
=95=95=95i=95=95=95X(=95)X1991-96  =95=95=95=95 =95=95=95=95=95=95=95 =95o=
=95=95=95=95=95=95[
   =C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=
=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=

[...]

#         DAT[                            XXXXXXXXXXXXXXXX
]                                         XXXXXXXXXXXXXXXX
#   i      DAT 2048 /* ........... XXXXXXXXXXXXXXiX  */[
#   i     0     ->      [                 XXXXXXXXXXXXXXXX
#   i     1     ->     1[                 XXXXXXXXXXXXXXXX
#   i     2     ->     2[                 XXXXXXXXXXXXXXXX
#   i     3     ->     3[                 XXXXXXXXXXXXXXXX
]                                         XXXXXXXXXXXXXXXX
            [] =3D " :\\    \\DAT\\    .DAT";X/*XXXXXXX */[
             0 {[                         XXXXXXXXXXXXXXXX
            [   DAT];[                    XXXXXXXXXXXXXXXX

[...]

           3; /*       */[         XXXXXXXXXXXXXXXX
        DAT [6]; /*         DAT     XXXX*/[XXXXXXXXX
           4[87]; /*       */[     XXXXXXXXXXXXXXXX
      o    ; /*  o    o       .  */[XXXXXXXXXXXXXXX
          0[6]; /*          o       0X*/[XXXX/*Xmenu macro keys 0 */
          0[26]; /*          o      X0X*/[XXX/*Xmenu macro name 0 */
          1[6]; /*          o       1X*/[XXXX/*Xmenu macro keys 1 */

[...]

) *=95=95=95;[                                 XXXXXXXXXXXXXXXX
=95=95=95=95=95=95 =95i=95=95=95 =95=95=95DAT =3D {[        XXXXXXXXXXXX=
XXXX
  =95=95=95=95=95=95=95, 0, 0, 0=95, 0=95, =95=95=95DAT, NULi[XXXXXXXXXXXXX
};[                                        XXXXXXXXXXXXXXXX
]                                          XXXXXXXXXXXXXXXX
#   i [                                    XXXXXXXXXXXXXXXX
]                                          XXXXXXXXXXXXXXXX
/* =CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=
=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD=CD   =
  */[
   resistance is fertile                   XXXXXXXXXXXXXXXX     PbN
```

Ausschließlich Maschinencode benützt auch Giselle Beiguelman in ihren *Code Movies*. Sie entnimmt hexa, ASCII, und binäre Codes aus Bildern im jpg-Format, speichert sie als Text ab und bearbeitet sie mit Hilfe des *Flash players* nach. Das Projekt weist auf die unsichtbaren Übersetzungsvorgänge im digitalen Medium hin und wirft die Frage nach der Rolle des *Source code* bei der Bedeutungskonstruktion auf.
Code Movie1

Ein Gedicht in Programmiersprache kann als Oberflächentext oder als Kette von Befehlen gelesen werden, zumindest wenn man als Leser Programmierkenntnisse mitbringt. In diesem Fall kann sich auch die ungefähren Auswirkungen der Befehle ausmalen, ähnlich wie man als Leser eines Drehbuchs oder einer musikalischen Partitur eine Vorstellung von der möglichen Realisierung entwickelt. Der poetische Gebrauch von Programmcode wirft die Frage nach den Unterschieden zwischen natürlicher und Maschinensprache auf, er verweist aber auch auf das prinzipielle Wirkungspotential jeder sprachlichen Äußerung. Und die Frage nach dem Wirkungspotential von Literatur kann der Literaturwissenschaft (und allen, die lesen oder schreiben) nicht gleichgültig sein. Außerdem kann man nie hundertprozentig sicher sein, dass

das ‘Gedicht’ nicht doch irgendetwas im Computer bewirkt, vom Umgang mit Viren und Würmern ist man ja unliebsame Überraschungen gewohnt.
Programmieren wird zuweilen auch als Kunst betrachtet, und warum eigentlich nicht? Auch die Produktion und die Verbreitung von Viren wird zumindest in Hackerkreisen als eine Art Kunst angesehen. Jedenfalls ist es nicht überraschend, dass Programmcode für künstlerische Zwecke verwendet wird.
Lyrik in Programmiersprachen existiert vereinzelt seit den sechziger Jahren. Auch hier hat das OuLiPo Pionierarbeit geleistet. Sein Manifest von 1962 fordert unter anderem, auf Programmiersprachen zurückzugreifen. 1972 verfasste Francois Le Lionnais Gedichte in Algol-Code, die sich zum Beispiel folgendermaßen lasen:

```
Table
Begin: to make format,
go down to comment
while channel not false
(if not true). End.
```

Ein Haiku von Larry Wall, dem Erfinder der PERL- Sprache (Practical Extraction and Report Language), die hier auch verwendet wird, lautet:

```
Print STDOUT q
Just another Perl hacker
Unless $pring
```

Liest man den Text laut (q = queue, $pring = dollar spring), erfüllt das Gedicht die Anforderungen an ein Haiku, nämlich Zeilen von fünf, sieben und fünf Silben aufzuweisen und eine Jahreszeit zu erwähnen. PERL ist nicht so weit entfernt von natürlicher Sprache, aber es verfügt nur über ein sehr beschränktes Vokabular von ca. 250 Wörtern, mit denen man als Dichter das Auslangen finden muss. Extreme Verdichtung ist das Markenzeichen des folgenden anonymen Liebesgedichts.

```
sub merge{
    my $enses;
```

Ein interessanter Aspekt der PERL-Gedichte ist der Umstand, dass sie tatsächlich als Programme laufen können und dann einen anderen (poetischen?) Text, den Maschinen-lesbaren Text, produzieren. Sie stellen also Palimpseste dar, was bekanntlich auch von Literatur im Lichte der Intertextualität behauptet wird. Man hat sogar vier Kategorien von PERL-Gedichten unterschieden: 1) nicht-funktionale, 2) als Programme ausführbare, 3) *keyword poems*, die ein PERL-Befehlswort als Ausgangspunkt benützen, und 4) humoristische *wordsalad poems*.

Mehr Raum für Kreativität bieten Texte, die natürliche und Computersprache mischen. Es entstehen dadurch hybride Texte oder Texte, die den strikt logischen Aufbau von Code als

Muster für poetische Sprache verwenden. Die natürliche Sprache wird sozusagen durch *Program code* kontaminiert.Das Augenmerk wird auf die Tatsache gelenkt, dass *Program code* in unseren Unterhaltungsmedien allgegenwärtig ist und insbesondere bei der Textproduktion maßgeblich beteiligt ist. Gleichzeitig fordern solche hybriden Texte die menschliche Wahrnehmung heraus und werfen die Frage nach der Grenze auf, bis zu der Texte lesbar bleiben. Im Hintergrund stehen Theoretiker einer posthumanen Kultur, die vorhersagen, dass unsere Kultur von einem Zeichensystem usurpiert wird, das nur auf unterschiedlichen elektrischen Ladezuständen beruht und dazu geeignet ist, Menschen und Maschinen zu 'vernetzen' und damit letztlich aneinander anzugleichen. Die spielerische *Program Code poetry* schwankt zwischen dem Akzeptieren des Unausweichlichen und dem Protest gegen die Mechanisierung der Kommunikation. Es ist gewissermaßen eine Demonstration des Siegeszugs der digitalen Technologie, wenn die poetische Sprache von Maschinenzeichen durchsetzt ist, wenn sich *dots* und *file extensions* einmischen wie *cyborgs* unter 'echte' Menschen, ein * eine beliebig zu füllende Leerstelle andeutet oder .tmp eine vorübergehende und später durch eine andere ersetzte Funktion bezeichnet.
Einfache Formen der *Code poetry* verwenden Zahlen und einige Zeichen und Symbole aus Computersprache als eine Art Versatzstücke, die Buchstaben oder Wörter ersetzen und so die natürliche Sprache verzerren und verfremden. Der Code und damit das für Laien äußerst geheimnisvolle Geschehen in den Rechnern wird zumindest teilweise sichtbar. Zutage treten dabei die in allen Sprachen vorhandenen autoritären Kontrollmechanismen. Es überrascht nicht, dass das hybride *Codework* sich oft mit Fragen von Identität und Gender sowie der Technologie und ihrem Einfluss auf das menschliche Verhalten beschäftigt und globalisierungskritische Töne anschlägt.

Mary-Anne Breeze (Mez) nennt das von ihr verwendete poetische Idiom "mezangelle" (mit Anspielung auf 'angel', 'mangle', 'angle', 'elle'). Sie beschreibt es als ein "polysemic language/code system [...] which evolved/s from multifarious email exchanges, computer code re:appropriation and net iconographs". Wörter werden manipuliert, auffällig sind insbesondere die häufigen Portmanteau-Wörter (Schachtelwörter), das Spiel mit Allusionen und Homophonen, das Einbauen von Symbolen, sowie von "tree structures, wildcard refs, boolcanisms, unix shell commands, bare html conventions, ascii][esque][tracks [as well as the repeated allusions 2 hyperlinks and html code via bracketing & directory slashings] - all act 2 illustrate the x.pansion of software potentialities of co:d][iscours][e". Kurz: "mezangelling attempts to expand traditional text parameters through layered/alternative meanings embedded in languages and the codes that create them." Eine andere Selbstbeschreibung lautet: "2 _mezangelle_ means to take words/wordstrings/sentences and alter them <in> such a way as to enhance meaning beyond the predicted or the expected. It is similar to making "plain" text hypertextual via the arrangement, dissection, and splicing of code scripting practices in2 english."
Das auffälligste Merkmal von "mezangelle", das Aufspalten von Wörtern, lädt zur Bildung neuer Silbenkombinationen und Bedeutungen ein und erinnert damit entfernt an *Finnegans' Wake*. Typisch für weite Bereiche der von Code kontaminierten Poesie ist der folgende selbstreflexive Text, der - wie viele Mez-Kreationen - in der Form eines e-mails oder E-Forum-Diskussionsbeitrags verfasst ist. Die ersten "mezangelle"-Texte sollen tatsächlich

Bearbeitungen realer mails gewesen sein. Hier ist es eine unter dem Pseudonym “hu][bris wo][man” firmierernde Schreiberin, die sich zum Thema “text|code text|code ::exe.][elo]cution text|code text|code” äußert.

.the
.randomness
.x.ists in the formulation

.coded][with][in their own

[hash d.finely ch[l]o[o]pped]
braille|small|emailness]

.semi-random but][ted [
.potentiali-T x.panse

% smaille editor = (

[omitted: lines of program code signs, composed of Xs, commas and various forms of brackets]

_
sparks of lost hu][bris][man §cent][+re.sieved][

Poetische Botschaften haben immer etwas Zufälliges, Beliebiges, die Netzkommunikation neigt zum Verkürzen und Verballhornen von Wörtern und bringt ein Idiom hervor, das für Uneingeweihte ähnlich fremd ist, wie die Blindenschrift. Code kann ausgeführt werden, das Subjekt, das sich als Hybris erweist, ist verloren, es bleiben nur schwache Funken davon übrig, die ‘resieved’ werden können. Neben den *dots* verwendet Mez hier hauptsächlich eckige Klammern, die in vielen Programmiersprachen Mehrdeutigkeit signalisieren. Eine ähnliche e-mail-Botschaft von “Phonet][r]ix” zum Thema “<][w][Rit][e][ua.LIS][P]tic>” handelt von “<SCRATCH Language=Juven][v]ilescrypt>”, was sich eventuell auf das Kratzen (Scratchen) von Schallplatten durch DJs bezieht und auf eine ramponierte Jugendsprache verweist, die der “Art Of][E][Go][a]-Blasting” dient. Ein weiteres Beispiel von “mezangelle” dreht sich um das Rollenspiel in *chat rooms* und den so genannten *multi user dungeons*.

:: Fantazee Genderator::> Assig.n[ation]inge Ov Charact.wh[m]orez 2
[w]Re[ck]quired Fiction.all.lie.sd Para.m[edical] Statuz

:: Vari.able[bodie]z::> Prince Cessspit N Princess Pit N Cin.der.ella[fitzgeraldingz] N Rap[t]punzelle N Gr.etal]

:: Will B Mild[h]er than me[aslez] but damaging to the fe[male]tus during

the first try[mester].

[5 Micah Dolls awai.ting AC.TIF.[f]ASHION]

Der spielerische *chat* wirkt als Fantasie-Gen(d)erator, der fiktionale Charaktere hervorbringt, die, weil man sie nur zeitweise benützt, mit Huren verglichen werden. Hier nehmen ein Prinz Cessspit, eine Prinzessin Pit, Cinderella (Fitzgerald), Rapunzel und Gretel teil. Der dritte Absatz scheint der Konversation entnommen zu sein, er könnte aber auch eine Selbstreflexion eines der Avatare sein.
Der Einbruch des *Program code* in die natürliche Sprache steht im Mittelpunkt von Mez' ".::...:the DataH Inpho[mill]ennium:.:..:", das von Christy Sheffield Sanford in ein Projekt mit dem Titel "'My' millennium", in dem verschiedene Künstler etwas zu diesem Thema beitrugen, aufgenommen wurde. Travestien von Bibelstellen und Gebeten, illustriert durch religiöse Motive, die durch Ketten von Binärcode verfremdet werden, läuten hier mit viel Ironie das neue Millennium ein, das eine schöne neue Welt der Information mit sich bringt, in der das $-Zeichen herrscht.

Then i sawe a C0mmandE @ the d0llah pr0mpte:
holdinge in itz teXt the keys of M: I: L: E: N: & U. 1t seiZed sum
0v the lettaHs: & K0dEd them up in2 a Th0usand N-crypti0nz. The
C0mmandE sent the N-crypti0nz staticscreaminge t0 the Binne.
The zer0Hs & w0nnes then flick.erred with joy, N the
DataHK0de was b0rnE with the re:maining lettaHz, the I: the
E: N 0therz B-sides, the P: & H: & O: all linked N clicking
straighte in2 the Inph0ennium...

Die Genesis wird hier mit dem Programmieren verglichen. Die Bibel, aber auch der Anfang des neuen Millenniums, des elektronischen Zeitalters, das durch die Götter der Berechnung (computation) eingeläutet wird, erscheinen in äußerst ironischem Licht. Die unterschiedlichen Bedeutungen von *command* (Befehl in der Computersprache, mit Anklang an *commandment* = biblisches Gebot) wird hier angesprochen, ebenso wie das Dollarzeichen, das in vielen Programmiersprachen verwendet wird und in *Program Code poetry* auf die Okkupation des Internet durch die Geschäftswelt aufmerksam macht. Computertechnik beruht auf "encryptions", auf Übersetzung in eine Programmiersprache, die ebenso geheimnisvoll ist wie manche Stellen der Bibel. Die übersetzten Texte werden an eine "bin(ne)" gesandt -laut Oxford English Dictionary "a receptacle" und insbesondere "each of a series of ranges of numerical value into which data are sorted in statistical analysis" -und dort der binären Umkodierung unterzogen, also n Ketten von Nullen und Einsern umgewandelt.

Ein anderer Teil des Textes ist mit "Pr0phetiCk-t0ck" überschrieben:

N we shell l00k 2 the [screen], N on those [screens] we shell d1nd the line----_---__
and the d0t:: . . , and the d0t shell speax. N the d0t will teXt:

$take up yr keys and type
$N click yr modemz :0N:
$N scroll the pages thru
$N delete @b0minations
$N n0.s N linkz that breakE the koDE
$N thenne merge the stringZ
$N u shell B phree.

Das Dollarzeichen, das bereits in der ersten Lexia vorkam, unterstreicht hier den 'prophetischen' Charakter von Quellcode. Die Hochstellung des Wortes "screen" verdeutlicht seine herausragende, fast kultische Stellung. Alles in allem zeichnet sich das Gedicht durch die Verarbeitung und Verfremdung aus dem Computerjargon entlehnter Wörter aus. Der Text, der zugleich auch einen Hypertext darstellt, schließt mit einer Hymne auf das Internet:

010010100Pray.err@//Praising the DataH- - -
DataH, let uz pray
2 the l0gin & the password:
the 1RC and the !CQ:
the 1Fs & Thennes:
the zer0Hs n the w0nnes:
Let us swhymn thru the netw0rks
N web into the fabr1c of yr f0nts:
Bin[ary]d typ0h and tract
in2 rhiz0mes 0v yr k0de.

Dazu noch ein Kommentar von Mary-Anne Breeze: "This period [.] of strategically driven hype[rtime] intersects perfectee with the cult of the Information Age - both in scope & paranoia. We seem 2 cult.u.rally invest similar amounts ov consumptive eNerGY in2 PMT [P/re/sent/post Millennium Tension] & hy[pah!]steria, as 2wardz our tandem ack!knopwledgement of technology as sa[tanic]viour and potenti.all religion re:placement."
Inphoennium

Ein außergewöhnlich umfangreicher Text, der *Program code* verarbeitet, ist *Lexia to Perplexia* von Talan Memmott, einem Medienkünstler und Designer. Das Werk besteht aus zehn verlinkten Webseiten, auf denen sich zum Teil programmierter Text befindet. Die ganze Installation ist instabil und irgendwie 'nervös', Texteinheiten verschwinden oder verändern sich oft schnell. Auch das ist ein Teil der Reflexion des Verhältnisses von User, Bildschirm und Netzwerken. Memmott vertritt die These, dass Mensch und Technik gleichberechtigt sind und einander im Verlauf der Geschichte gegenseitig hervorbringen. Körper werden mit elektronischem Material gleichgesetzt, beide sind daher starken Wandlungen unterworfen. Die Begegnung von User und Rechner führt zu der im Titel genannten Perplexia. In einem Interview erläutert Memmott, dass er darunter "a confusion of ontological, literary and technical application" versteht. Die Texte bewegen sich an der Grenze der Lesbarkeit, was aber nur folgerichtig ist, wenn Mensch und Maschine, Fleisch und elektronisches Material

miteinander kommunizieren. Immer wieder kehrt das Begriffspaar Head - Body. Gleich im *opening screen* des ersten von vier Kapiteln findet sich die Zeile:

<HEAD>[FACE]<BODY>, <BODY> FACE </BODY>

Hier ist sowohl von Körperteilen als auch -als HTML-Code gelesen -von Textteilen die Rede; übersetzt bedeuten die zwei Phrasen Face ist Teil von Head, aber nicht von Body bzw. Face ist Teil des Körpers. Head und Body sind aber auch wesentliche Bestandteile eines Quelltextes. Der Quelltext zur Lexia Anonymus [N] lautet zum Beispiel:

```
<HTML>
<HEAD>
<TITLE>Anonymus.[N]</TITLE>

<script language“JavaScript“>
<!—
function loaded() {
if (parent.location.href.index0f(“FRAME.html” != -1)
parent.update(); // only called when inside preloader frameset
}
// →
</script>
<style>
body {background-color:#000000;font-
family:courier,arial,helvetica; font-size 10pt}
A{color:#336033; text-decoration:none;}
A:hover{color:#FFFFFF}
A:visited:{color:#c0c0c0;}

…
```

Alles, was zwischen <HEAD> (= opening tag) und </HEAD> (= closing tag, hier nicht mehr enthalten) steht, gehört sozusagen zum Head.
Derselben Beziehung widmet sich auch ein Song, der wiederum natürliche und Programmiersprache vermengt. Ein Satz darin lautet:

```
I will conceal (to you, my dear) {
(this body.skin) {
```

Das kann als syntaktisch etwas eigentümlicher, aber noch verständlicher englischer Satz durchgehen. Vermutlich will das sprechende Ich seine Haut vor anderen (my dear) verbergen. Es folgen mehrer Umformungen des Satzes, in denen das Ich sagt, dass es verwundbar (vulnerable) ist, sich schließlich aber doch offenbart (reveal). Die Textstelle enthält aber auch *Program code*-Elemente, wobei der Punkt zwischen Body und Skin bedeutet, dass das Objekt

Body die Eigenschaft Skin besitzt. Es fehlt allerdings noch eine Anweisung, zum Beispiel - beliebig gewählt - die Zuweisung einer Farbe: body.colour=blue. Memmott verwendet Javascript und DHTML, es handelt sich aber um Pseudocode, der zwar an diese Sprachen erinnert, aber eben nicht ganz korrekt ist.
Wie bei Mez manifestiert sich der *Program code* nicht zuletzt in der Verfremdung von Wörtern, wie zum Beispiel “cell.f” oder “cell…(f)” für “self”. Der User, das Ich, wird häufig als “I-terminal” bezeichnet (auch:eye-terminal) und dem“ x-terminal” (=computer) gegenübergestellt. I/eye ist ein wichtiger Gesichtspunkt, da immer wieder digitalisierte Augen am Bildschirm erscheinen, die uns gewissermaßen aus dem Inneren des Bildschirms betrachten, aber auch Reflexion unserer eigenen Augen sein könnten, also gegenseitige Betrachtung anzeigen.
Anspielungen auf Kunst, Literatur und Mythen durchziehen den Text. Freud, Nietzsche, Heidegger, Deleuze und Guattari sind häufigeReferenzen. Eie Anspielung auf Freuds *Unbehagen in der Kultur* (Civilisation and ist Discontents) steckt zum Beispiel in der Zeile ”Cyborganization and its Dys ı Content(s)”. Freuds Behauptung, der Mensch strebe danach, mit der Mutter zu einer Einheit zu verschmelzen, wird auf die digitale Technologie übertragen, die mit der Mensch verschmilzt und die nun als ein notwendiger, aber verhängnisvoller Kulturprozess erscheint. Der Mythos von Narziss und Echo wiederum wird im Zusammenhang mit den Augen herbeizitiert, die als Augen des Betrachters, der sich somit selbst bespiegelt, interpretiert werden können. Für die postmoderne Literaturtheorie von Bedeutung ist auch der Umstand, dass Echo nur wiederholen kann, was andere gesagt haben. Auch *Lexia to Perplexia* ist durchzogen von solchen Wiederholungen und Echos. Schließlich wird auch der ägyptische Isis und Osiris-Mythos bemüht. Memmott erläutert in dem bereits erwähnten Interview, dass Osiris auch Ausere genannt wird, was nun wiederum als “A User” gelesen werden kann. Über diesen Mythos kehrt die Thematik des Körpers, seines Verhältnisses zur Seele, von Tod, Einbalsamierung und Sublimation (Vergeistigung) wieder, die die Auferstehung des Users als virtuelles Techno-Subjekt impliziert.
Lexia to Perplexia