c/o Freie Universität Berlin
Seminar für Allgemeine und Vergleichende Literaturwissenschaft
Hüttenweg 9
14195 Berlin

SUB MERGE {MY $ENSES; ASCII ART, REKURSION, LYRIK IN PROGRAMMIERSPRACHEN

FLORIAN CRAMER

INHALTSVERZEICHNIS

Bislang wurden vor allem solche elektronische Texte „Netzliteratur“ genannt, die experimentelle Erzähl- und Schriftformen der literarische Moderne (und Postmoderne) von der Buchseite auf den Computerbildschirm transponieren und Kontinuitäten und Differenzen erkunden, die diese Transposition zeitigt. Wenig beachtet wurden hingegen poetische Spielformen, die von Programmierern und Nutzern des Internets entwickelt wurden lange bevor der Begriff „Netzliteratur“ entstand. Dieser Text soll einige von ihnen untersuchen.

Ein historischer Rückblick: Zwar existiert das Internet, einschließlich seiner Prototypen, bereits seit über dreißig Jahren, doch erst als um 1993 das World Wide Web erfunden wurde und die Browser *Netscape Navigator* und später *Microsoft Internet Explorer* einen Standard für Bildschirm-Anzeige elektronischer Texte etablierten, entstand eine „Netzliteratur“, die ihren Namen als de-facto-Synonym von „Hypertext“-Dichtung bzw. „Hyperfiction“ begriff.[1] Das Konzept des „Hypertexts“ und der Netz-„Hyperfiction“ umfaßt assoziative Verknüpfungen von Texten, die auf einem oder auf verschiedenen Computern verstreut sind, setzt aber keine Programmierung voraus. Statt den Computer selbst zu programmieren und ihn Text generieren zu lassen, wie es Ende der 1950er Jahre und Anfang der 1960er Jahre die Computerlyrik des *Beat poet* Brion Gysin, der Stuttgarter Gruppe um Max Bense und der französischen Oulipo-Dichter um Raymond Queneau und François le Lionnais, [2] nutzt die neuere „Netzliteratur“ den PC mit seiner vorinstallierten Software als vernetztes Bildschirmlesegerät.

Date: 5.2.2001.

[1]Siehe z.B. Michael Böhlers und Beat Suters Sammelband *Hyperfiction.* Hyperliterarisches Lesebuch: Internet und Literatur [SB99]

[2]Vgl. [Gys78], [Döh98], [Mol71] und [Fou77]

Nicht nur in dieser „Netzliteratur“ selbst, sondern auch in ihrer Kritik und Philologie wurden poetische Formen übersehen, die seit den 1970er Jahren in andere Netzkulturen entwickelt wurden, ohne explizit als literarisch ausgewiesen zu werden. An ihnen zeigt sich, wie problematisch es überhaupt ist, aus dem Internet und der Computerkultur ein spezifisch literarisches Feld zu destillieren und von Nicht-Literatur zu unterscheiden. Denn sogenannte Softwarearchitekturen, das Internet eingeschlossen, sind selbst nichts als Text;[3] Text, der in Computersprachen geschrieben ist und als digitaler Schriftcode übermittelt, transformiert und ausgeführt wird. Auch vermeintlich multimediale Bild- und Tondaten sind, bis sie den Rechner verlassen, Textcode und als solche mit rhetorischen Operationen der Wiederholung, Auslassung und Verschiebung manipulierbar. Da das Internet ein Textgebilde aus den komplex gewebten Codes von Betriebssystemen, Hilfsprogrammen, Programmiersprachen und Netzwerkprotokollen,[4] von denen „Hypertext“ und „World Wide Web“ nur eine äußerste und arbiträre Repräsentationsschicht sind, doch nur wenige sogenannte „Netzliteratur“ diese Struktur halbwegs kennt und reflektiert, ist die Schriftsteller-Avantgarde des Internets bei seinen Programmierern zu suchen. Seitdem im Jahre 1969 an der University of California at Los Angeles der erste Knoten des späteren Internets in Betrieb genommen wurde, sind Systemprogrammierer und Netzwerkadministratoren, aber auch Hacker, die den Protokollcode verletzen und subvertieren, die Schreiber mit dem komplexesten Verständnis digitaler Schriftlichkeit. Ihre Kultur hat eigene poetische Spiele entwickelt, von denen wenige wie die bildschirmtypographischen „Emoticon“-Smilies populär geworden sind,[5] und andere hier vorgestellt werden sollen: ASCII Art, rekursive Akronyme und Lyrik in Programmiersprachen sind nicht nur bemerkenswerte poetische Formen an sich, sondern werden in der konzeptkünstlerischen *Net.Art* fortgeschrieben und interessant modifiziert.[6]

ASCII ART

ASCII, der „American Standard Code for Information Interchange“, ist seit 1963 der kleinste gemeinsame Nenner aller Computer-Systemschriften: ein Alphabet von 128 Buchstaben, Zahlen und Interpunktionszeichen, das dem Zeichenvorrat einer amerikanischen Schreibmaschine entspricht und deshalb weder Umlaute, noch andere spezifische Zeichen nichtenglischer Sprachen enthält.[7] ASCII-Text ist Computertext ohne jegliche typographische Auszeichnungen, ohne Schriftformatierungen also, und bildet nach wie vor den Standardcode für E-Mail, sowie das Zeichenrepertoire aller Programmiersprachen und der meisten Netzwerkprotokolle. Bevor die graphischen Benutzeroberflächen unter anderem des Apple Macintosh und von Microsoft Windows Graphiken und formatierten Text einführten und bevor das World Wide Web diese Neuerung aufs Internet übertrug und auch

[3] d.h. Steuerungen von Systemzuständen, die als Text gelesen werden können (so, wie z.B. auch Rauch-, Flaggenzeichen oder Schachpositionen als Text lesbar sind) bzw. linear, vollständig und fehlerfrei in alphanumerischen Text konvertiert und aus ihm in Systemzustände rückkonvertiert werden können.

[4] Vgl. [Les00]

[5] wie :-) und :-(als um 45 Grad gedrehte Typogramm-Gesichter, die Freude und Enttäuschung ausdrücken.

[6] S.a. die Anthologien [Bau99], [WD00] sowie vor allem Valentina Djordjevics Artikel *Textverarbeiter und Screendesigner* [Djo99]

[7] Eine ausführliche Geschichte des ASCII-Standards schreibt Dieter E. Zimmer in seinem Aufsatz [Zim97].

beliebige Dateien als E-Mail-Anhang verschickt werden konnten, spielte sich alle Netzkommunikation im ASCII-Code ab. Dort, wo dies als Mangel empfunden wurde, entstanden analog zu mechanisch getippten Schreibmaschinen-Graphiken elektronische ASCII-Typogramme, die Fotographien und Zeichnungen (oft Comic-Illustrationen) durch Buchstaben und Zahlen nachbilden. Im Gegensatz zu den Ideogrammen der konkreten Poesie und den Figurengedichten der Antike und frühen Neuzeit sind Buchstaben und Zahlen in der ASCII Art nur visuelle Raster, die allenfalls in Ausnahmsfällen auch schriftsprachlich semantisiert werden. Die ASCII Art des frühen Internets dient ausschließ als piktographischer Bildträger. Ihre naive Bild-Mimesis, aber auch die heutigen Ideologien des Multimedialen ironisiert der Text-Film *Deep ASCII* des *ASCII Art Ensemble*, dessen Kern die Netzkünstler und Programmierer Vuc Cosic und Walter van der Cruijsen bilden.[8] *Deep ASCII* digitalisiert den Pornofilm-Klassiker *Deep Throat* (1972) zu einem bewegten Zeichenraster auf dem Computerbildschirm; Personen und Schauplätze sind, mit einigen Metern Abstand betrachtet, noch als bewegte Schemen auf der grün-schwarzen Textanzeige erkennbar. Der Effekt dieser Umcodierung ist paradox; einerseits treibt sie der Pornographie durch abermalige Schematisierung ihre Obszönität aus, andererseits gibt die Unklarheit der Bildraster der Einbildungskraft Freiräume zurück und reauratisiert das Gezeigte obszön. Mit der Linux-Software *AAlib* und *Aview* der Programmierer Jan Hubicka, Kamil Toman, Thomas A. K. Kjaer und Tim Newsome sowie dem Programm *HASCIICAM* von Denis Roio können mittlerweile beliebige Graphikdateien als Rastertexte gelesen und auch Video und Fernsehen im ASCII-Format betrachtet werden.[9]

Liest man die Texte der ASCII Art als Literatur, so sind sie im Gegensatz zur klassischen Figurendichtung zwar völlig kontingent und bar jeder semantischen Verdichtung. Ihre Poetik aber ist insofern auch eine literarische, als sie die textuelle Codiertheit aller digitalen Zeichen reflektiert. Da in der ASCII Art die Codierung identisch ist mit der sinnlichen Repräsentation, sie als Bild wie als den Betrachter jedoch irritiert, zeigt ASCII Art, daß Codierung und sinnliche Repräsentation eben nicht gleich sind, und sich der Code vor dem Benutzer verbirgt. Kommerzielle PC-Betriebssysteme behandeln ASCII-Text nur noch als Rauschen oder Funktionsstörung. Auf Windows-PCs markiert er Betriebsmeldungen der Hardware und als „Blue Screen of Death“ den Systemabsturz, der Apple Macintosh hat ihn vollständig eliminiert. Die neuere, experimentelle ASCII Art aus dem Umfeld des netzkünstlerischen E-Mail-Forums *7-11* kehrt diese Rhetorik um, indem sie im Text gefundener Systemmeldungen und Netzwerkprotokolle Spuren von Fehlcodierungen, des Rauschens und der Systemabstürze auffindet und ästhetisiert. Ein gutes Beispiel dieser neueren ASCII Art ist der (hier nur in Ausschnitten wiedergegebene) Text „DAT CiDE NULi: resistance is fertile“ des kanadischen Netzkünstlers Ted Warnell:

```
                                  XXXXXXXXXXXXXXXX
/*     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[
=95o=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=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4=C4[
=95o=95=95 =95=95=95=95=95=95=95=95=95=95 =95=95=95=95=95=95=95=95=95   =
```

[8] ASCII Art Ensemble Homepage `http://www.desk.org/a/a/e/first.html`

[9] Web-Seite und Bezugsquelle der Software: `http://ascii.dyne.org`

```
=95o=95=95=95i=95=95=95X(=95)X1991  =95=95=95=95 =95=95=95=95=95=95=95 =
=95o=95=95=95=95=95=95[
    [...]
           i       DAT =3D {[           XXXXXXXXXXXXXXX
, 0, 0, 0 , 0 ,    DAT, NULi[XXXXXXXXXXXXX
};[                                 XXXXXXXXXXXXXXX
]                                   XXXXXXXXXXXXXXX
#   i [                             XXXXXXXXXXXXXXX
]                                   XXXXXXXXXXXXXXX
#i        DAT[                      XXXXXXXXXXXXXXX
#         DAT[                      XXXXXXXXXXXXXXX
]                                   XXXXXXXXXXXXXXX
#   i      DAT 512 /* ...........  oXXXXXXioXXXXXXX  i   */[
#   i     0    ->      [           XXXXXXXXXXXXXXX
#   i     1    ->     1[           XXXXXXXXXXXXXXX
]                                   XXXXXXXXXXXXXXX
[] =3D "\\   \\   .DAT"; /*XXoXXXXXXioXX*/[
0 {[                    XXXXXXXXXXXXXXX
[   DAT];[              XXXXXXXXXXXXXXX
    [...]
             3[6]; /*          o      3X*/[XXXX/*Xmenu macro keys 3 */
3[26]; /*          o     X3X*/[XXX/*Xmenu macro name 3 */
4[6]; /*          o      4X*/[XXXX/*Xmenu macro keys 4 */
4[26]; /*          o     X4X*/[XXX/*Xmenu macro name 4 */
5[6]; /*          o      5X*/[XXXX/*Xmenu macro keys 5 */
5[26]; /*          o     X5X*/[XXX/*Xmenu macro name 5 */
6[6]; /*          o      6X*/[XXXX/*Xmenu macro keys 6 */
6[26]; /*          o     X6X*/[XXX/*Xmenu macro name 6 */
7[6]; /*          o      7X*/[XXXX/*Xmenu macro keys 7 */
7[26]; /*          o     X7X*/[XXX/*Xmenu macro name 7 */
[128]; /*       */[    XXXXXXXXX/*Xblank */
};[                                 XXXXXXXXXXXXXXX
=95=95io=95 =95=95=95_=95=95=95o=95=95 {[               XXXXXXXXXXXXXXX
=95=95=95=95=95=95 =95=95=95=95=95=950 =95=95=95=95=95=95;[          XXXX=
XXXXXXXXXXX
=95=95=95=95=95=95 =95=95=95=95=95=951 =95=95=95=95=951;[          XXXXXX=
XXXXXXXXX
} *=95=95=95;[                       XXXXXXXXXXXXXXX
=95=95=95=95=95=95 =95i=95=95=95 =95=95=95DAT =3D {[           XXXXXXXXXXX=
XXXX
=95=95=95=95=95=95=95, 0, 0, 0=95, 0=95, =95=95=95DAT, NULi[XXXXXXXXXXXX
};[                                 XXXXXXXXXXXXXXX
]                                   XXXXXXXXXXXXXXX
#   i [                             XXXXXXXXXXXXXXX
]                                   XXXXXXXXXXXXXXX
/* =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           XXXXXXXXXXXXXXX     PbN
```

Indem der Titel *Raumschiff Enterprise*-Dialogzeile „you will be assimilated – resistance is futile“ travestiert, weist er sich als Produkt einer Netzpoesie aus, die ASCII-Code und üblicherweise verdeckte und gefilterte Protokollschichten der Computernetze als subversives Zeichensystem begreift. Hierin folgt die Netzkunst der Netz-Subkultur jugendlicher Software-Raubkopierer und Angreifer von Netzwerkrechnern, die sich in einer eigenen, teilweise durch Softwarefilter erzeugten ASCII-Codesprache verständigen; eine Sprache, in der z.B. die respektvolle Bezeichnung eines Raubkopierers „7331 WAR3Z dOOd“ („leet [=elite] wares dood“) lautet.

Der ASCII-Poesie der Absturzcodes geht es, im Gegensatz zur klassischen Figurendichtung, nicht um synthetische Konstruktion, sondern um das analytische Auffinden, Umcodieren und Wiedereinspeisen von Zeichenmaterial. Das unterscheidet sie auch von den „Konstellationen“ der konkreten Poesie, mit denen sie ansonsten das Merkmal des visuell Amimetischen teilt, und rückt sie in die Nähe dadaistischer und lettristischer Buchstabenmontagen. Die ASCII Art gewinnt im Moment, da sie von der Bild-Mimesis zu einem rein selbstbezüglichen Spiel mit Computerzeichen wird, an disruptiver Qualität, indem sie die Differenz von Steuercode und Repräsentation wieder unkenntlich macht. In einem Medium, das wie das Internet selbst aus Code konstruiert ist, sind ihre Störzeichen tatsächlich als technische Störung interpretierbar. Ihre Absender potenzieren diesen Effekt dadurch,

daß sie ASCII-Montagen wie unerwünschte Werbe-E-Mail massenhaft als „Spam-Art“ verschicken und auf Verteilerlisten wie *7-11* in Kreislauf perpetueller Recodierung speisen.

Die Website des holländisch-belgischen Duos Jodi `http://www.jodi.org` wurde sogar zwangsweise vom Netz entfernt, als Systemoperatoren ihre Störsimulationen für einen Virus hielten. Seit Mitte der 1990er Jahre gehören Jodi zu den international bekanntesten zeitgenössischen Netzkünstlern. Ihre Arbeit *Location* zeichnet beispielhaft die Schwelle von der bildmimetischen zur amimetisch-disruptiven ASCII Art auf. Was im Fenster des Webbrowser wie Störzeichen aussieht (Abb. **??**), ist im Quellcode ein traditionelles ASCII-Kitschbild eines Teddybärs (Abb. **??**). Die Webbrowser-Software selbst codiert, ohne weitere Programmierung, die Teddybären in ein nichtmimetisches Schriftbild um, da sie die Zeilen des Typogramms neu umbricht und sein Raster damit auflöst. Die ästhetischen Erwartungen an digitale Zeichen, in der formatierten Repräsentation „lesbar“ und im Quellcode „unlesbar“ zu sein, wird hier auf den Kopf gestellt. Indem die Arbeit mit ihren sichtbaren und verborgenen Zeichen spielt, verlangt sie einen wissenden und neugierigen Leser, der den technischen Unterschied von Anzeige und Quellcode überhaupt kennt und beide Zeichenebenen zu entziffern weiß.

Viele Textspiele mit der Selbstbezüglichkeit digitaler Codes führt zu Texten sind nur noch für Computerkenner lesbar. In einem Gedicht des amerikanischen Essayisten Alan Sondheim bedichtet sich das Textbearbeitungsprogramm, mit dem es geschrieben wurde – die unter Unix und Linux verbreitete Minimalsoftware „vi“ – selbst:

```
vi~
vi~
vi~
vi~
vi~sor
vi~
vi~the cursor pauses (cursor moves here) cursor makes a path (cursor says
vi~i'm here) cursor wanders makes a path (cursor says this is my field)
vi~cursor says this is my forest (my mountain crag) my rocky stream (cur-
vi~sor meanders makes a path) cursor was here (cursor paused here) cursor
vi~was lost here (here the words were saved) here they moved again (cursor
vi~left and returned) cursor left and paused (here cursor left) here cur-
vi~sor paused (mounty crags) (rocky streams) (twisty paths) cursor pauses
vi~here
vi~
vi~
vi~
vi~
vi~
```

Die Typographie des Texts entspricht dem Bildschirmfenster von „vi“, und seine Wörter beschreiben die Operationen der Texteingabe, die Bewegungen der Textmarke, die mit zum Schreiber spricht, zwischendurch auch die Sicherung der Datei. Da das Gedicht einen Schreibakt inszeniert, der sich im Moment seines Geschehens schon selbst beschreibt, konstruiert er etwas unmögliches und erweist sich als dichterische Simulation. Auch Sondheims Text zeigt unterschlagenen Code auf, jenen Text nämlich, den jeder Computerschreiber eintippt, ohne daß er in der Schrift erschiene; und wie die ASCII Art ästhetisiert er den Steuercode zu avantgardistischer Poesie.

REKURSION

Die Eigenschaft digitaler Daten, mindestens doppelt lesbar zu sein – als Steuercode und als formatierte Repräsentation –,[10] macht sich die experimentelle Netzkunst zu Nutze, indem sie beide Zeichenebenen miteinander kontaminiert, rückkoppelt und verschleift. Ein Prozeß, der sich selbst aufruft, indem er seinen Output wieder in den Input speist, ist, computerwissenschaftlich gesprochen, eine Rekursion. Rekursiv ist zum Beispiel die Erzählung des chinesischen Philosophen Chuang Tzu, der träumt, ein Schmetterling zu sein und nach dem Aufwachen nicht mehr weiß, ob er ein Schmetterling ist, der träumt, Chuang Tzu zu sein;[11] rekursiv ist auch das *Frametale* aus John Barths *Lost in the Funhouse*, ein Möbiusband mit dem Satz „ONCE UPON A TIME THERE WAS A STORY THAT BEGAN“, rekursiv ist schließlich das Lied vom Mops in der Küche, das Becketts *Warten auf Godot* zitiert.[12] Douglas R. Hofstadter weist Rekursionen unter anderem in Bachs Fugen nach,[13] das modifizierte Kreter-Paradox „Dieser Satz ist eine Lüge“ ist eine logische Rekursion.

Trotz der Gefahr des „regressus ad infinitum“, der Schleife ohne Ausgang, ist Rekursion ist eine zulässige und verbreitete Methode in der Computerprogrammierung: Ein Abschnitt eines Programms (der z.B. ein vorgegebenes Wort um ein Fragezeichen ergänzt) schließt mit einem Aufruf seiner selbst und durchläuft sich so oft, bis eine bestimmte Abbruchbedingung (z.B. drei angehängte Fragezeichen) erfüllt ist. Als intellektuelles Spiel waren Rekursionen besonders unter den Systemprogrammierern am Artificial Intelligence Lab des MIT beliebt, die in den frühen 1970er Jahren eines der ersten Netzwerkbetriebssysteme schrieben und für sich das Wort „Hacker“ erfanden.[14] MIT-Hacker übertrugen rekursive Schleifen von Programmiersprachen auf die Umgangssprache, indem sie rekursive Akronyme für ihre Programme erfanden. Das Textbearbeitungsprogramm *EINE*, das als Alternative zum (auf Unix-Computern verbreiteten) Textprogramm *Emacs* entworfen wurde, ist eine Abkürzung von „EINE Is Not Emacs“.[15] Da die das Akronym sich in seiner Auflösung wieder selbst enthält, schreibt es sich unendlich fort:

(EINE)
= ([EINE] Is Not Emacs)
= ([{EINE} Is Not Emacs] Is Not Emacs)
= ([{<EINE> Is Not Emacs} Is Not Emacs] Is Not Emacs)
...

Der Nachfolger von *EINE*, *ZWEI*, steht für „ZWEI Was EINE Initially“. Seine Rekursion ist daher eine doppelte und führt zu noch komplexeren Textwucherungen:

(ZWEI)
= ([ZWEI] Was [EINE] Initially)
= ([{ZWEI} Was {EINE} Initially] Was [{EINE} Is Not Emacs] Initially)

---

[10] Der Kasseler Semiotiker und Kommunikationswissenschaftler Guido Ipsen untersucht diese Hybridität digitaler Zeichen in einem Dissertationsprojekt

[11] [Tzu64], S.45

[12] [Bar68], S.3

[13] Hofstadter, *Gödel, Escher, Bach* [Hof79].

[14] soziologisch beschrieben in [Tur84], Kap.6, S.241-294

[15] zitiert nach dem Eintrag „recursive acronyms“ des *jargon file* [rayoJ]

= ([{<ZWEI> Was <EINE> Initially} Was {<EINE> Is Not Emacs} Initially] Was [{<EINE> Is Not Emacs} Is Not Emacs] Initially)
...

Im Gegensatz zu den Erzähl-Rekursionen von Beckett und John Barth verschleift sich im poetischen Spiel der rekursiven Akronyme das bloße Wort; es wird zu einem Prozeß, der sich selbst steuert, und übersteuert. Die Selbstausführbarkeit von digitalem Programmcode überträgt sich hier in die Umgangssprache, das Wort wird zu einem sich selbst replizierenden Virus. Darin gleichen rekursive Akronyme zwar älteren sprachkombinatorischen Dichtungsformen wie dem buchstabenwechselnden Anagramm und dem wortwechselnden Proteusvers,[16] doch ist ihre Kombinatorik nicht nur scheinbar, sondern auch faktisch unbegrenzt und katastrophisch. Mit jedem Prozeßdurchlauf wächst der Text des rekursiven Akronyms exponentiell an, und indem sich seine Selbstbeschreibungen immer komplexer verschachteln, verkompliziert sich auch die Logik des Satzes.

Die perfekt geschlossene, aber nicht minder selbstbezügliche Gegenform dieser katastrophischen Programmierung ist der sogenannte *Quine*, ein Programm, das eine Kopie seiner selbst erzeugt. Sein Steuercode muß so geschrieben sein, daß er sich seblst reproduziert.[17] Gary P. Thompsons *Quine Page* `http://www.nyx.net/~gthompso/quine.htm` verzeichnet *Quines* in dutzenden Programmiersprachen wie BASIC, C, Java, LISP, Pascal und Perl. Nichtprogrammierern am leichtesten verständlich ist ein *Quine* in der Programmiersprache BASIC:[18]

```
10 DATA "B$='DATA '+CHR$(34)
20 DATA "FOR J=10 TO 180 STEP 10
30 DATA "READ A$
40 DATA PRINT J;B$;A$
50 DATA "IF J<>90 THEN 170
60 DATA "RESTORE
70 DATA "B$=' '
80 DATA "NEXT J
90 DATA "END
100 B$='DATA '+CHR$(34)
110 FOR J=10 TO 180 STEP 10
120 READ A$
130 PRINT J;B$;A$
140 IF J<>90 THEN 170
150 RESTORE
160 B$=' '
170 NEXT J
180 END
```

Die Zeilen 10-90 schreiben den Steuercode der Zeilen 100-180 in einen Speicher, die Zeilen 100-180 drucken diesen Speicher in einem ersten Schritt so modifiziert aus, daß er Zeile 10-90 wiedergibt, und in einem zweiten Schritt ungefiltert, so daß er Zeile 100-180 reproduziert. Wird das Programm aufgerufen, erzeugt es also wieder den oben abgedruckten Steuercode, der wiederum als Programm gestartet werden kann, und so weiter. „Iterative Quines“ verfeinern dieses Prinzip: Ein Programm A erzeugt ein von ihm verschiedenes Programm B, das ein weiteres Programm C erzeugt, dessen Output wiederum Programm A ist. Das Konzept sich selbst reproduzierender technischer Systeme geht auf John von Neumanns Automatentheorie von 1951 zurück; doch schon Poetiken und Enzyklopädien sind rekursive Wissenssysteme, die dem Leser die Instrumente zur Replikation ihrer selbst

[16] Vgl. [Wag71]

[17] Benannt nach W.V. Quines logischem Paradox: „Ergibt etwas Falsches, wenn an sein eigenes Zitat angehängt“ ([Qui62], vgl. [Var81]), zitiert nach dem *jargon file* [rayoJ]

[18] von Donald Bell, Quellcode von der Quine Page `http://www.nyx.net/~gthompso/self_bas.txt`

in die Hand geben.[19] Selbstreplizierende Software wie die *Quines*, aber auch Computerviren, verbinden beide Aspekte; sie übersetzen die Wissensrekursion der Enzyklopädien in Automatenprozesse und die selbstreplizierenden Automaten in Text. Nicht nur schließen rekursive Programme, wie das visuell-poetische Spiel der ASCII Art, Codierung und sinnliche Repräsentation des digitalen Zeichens miteinander kurz; als simultane Partituren und Ausführungen dieser Partitur sind sie zugleich Poesie und die Poetik dieser Poesie.

LYRIK IN PROGRAMMIERSPRACHEN

Wenn *Quines* und andere selbstbezüglich-ironische Steuercodes die Frage aufwerfen, ob gewisse Spielformen der Computerprogrammierung auch als Lyrik interessant sind, bliebe umgekehrt zu klären, ob Gedichte auch als Computerprogramme eingesetzt und ausgeführt werden können. Die Idee, Lyrik in Programmiersprachen zu schreiben, wurde bereits in den 1960er Jahren von der französischen Oulipo-Gruppe entwickelt und, unabhängig davon, um 1991 von Programmierern im Internet popularisiert. Oulipo, die „Werkstatt für potentielle Literatur“ gründete sich 1960 in Paris. Ihre Poetik wurde von Raymond Queneau, dem Verfasser der kombinatorischen *Hunderttausend Milliarden Sonette*, und François le Lionnais geprägt, einem Mathematiker, der die zwei Manifeste der Gruppe verfaßte. Zwar lehnte Queneau den Begriff „experimentelle Literatur“ ab und gab den oulipotischen Dichtungsspielen den ironischen Gestus einer Sprachpataphysik, doch entwirft bereits das erste Oulipo-Manifest von 1962 ein konzises Programm einer Computerdichtung. Man beabsichtige, so heißt es darin, „bei Bedarf auf die guten Dienste von Datenverarbeitungsmaschinen zurückzugreifen“ und die „Kettentheorie von Markow“ zu verwenden,[20] die bis heute die Basis vieler poetischer Automaten – einschließlich Ray Kurzweils *Cybernetic Poet* – bilden. Das Manifest plädiert außerdem für poetische „Vorstöße“ auf das „Gebiet besonderer Vokabularien (wie denen von Raben, Füchsen, Tümmlern; die Programmiersprache von Computern – Algol – etc.)“.[21] Algol („Algorithmic Language“) wurde 1959 erfunden, und weil diese Computersprache gegenüber anderen ein besonders kleines Vokabular charakterisiert, eignete sich besonders als oulipotisches „contrainte“, als selbsterauflegte formale Restriktion wie jene, einen Text ohne „e“ zu schreiben.[22] 1972 schrieb Le Lionnais, und kurz nach ihm Noël Arnaud, Lyrik in Algol-Code, die als traditionelle Wortdichtung lesbar sein sollte:

xxxx [Lionnais, Algol-Gedicht][23]

Le Lionnais’ Gedicht erweitert Rhetorik und Poetik des Programmierens um eine neue Spielart – die des simulierten Programmcodes. Das Gedicht folgt zwar der Syntax von Algol, erzeugt, sobald man versucht, es als Programm auszuführen, aber nur eine Reihe von Fehlermeldungen.[24]

---

[19] Wau Holland, Meinungsfreiheit - das wichtigste Grundrecht `www.trend.partisan.net/trd1098/t021098.html`: „Die erste rekursive Sammlung des Wissens war die Enzyklopaedie von Diderot und d’Alembert. Prompt wurde sie vom Papst verboten. Da sie aber ein Rezept zum Nachbau von sich selbst enthielt, also genaue Beschreibungen, wie Kupferstiche gemacht werden, wie eine Setzerei und eine Druckerei aufgebaut sind, wie Erz geschmolzen wird usw., war das Verbot der Enzyklopaedie nur zehn Jahre durchsetzbar.“

[20] [BK93], S.20

[21] [BK93], S.21

[22] Diese „contrainte“ liegt Georges Perecs lipogrammatischem Roman „La disparition“ zugrunde.

[23] [MB98], S.47

[24] Daß simulierter und fehlerhafter Programmcode als eigene Form digitaler Kunst und Poesie anerkannt werden sollte, zeigte sich auch in den Diskussionen und im Abschlußmanifest der Jury des Preises für künstlerische Computersoftware, den das Berliner *transmediale*-Festival erstmals im Jahr 2000 ausgeschrieben hatte.

Vorgetäuschte und echte Ausführbarkeit von Programmcode charakterisiert auch die *Perl Poetry*, die zur zeitgenössischen Nachfolgerin der oulipotischen Algol-Lyrik geworden ist. Notiert ist sie in der Programmiersprache Perl („Practical extraction and report generation language“), die von dem Computerlinguisten Larry Wall bewußt nah an der Umgangssprache entwickelt wurde, einen großen Umfang von Wortbefehlen besitzt und Programmierern große Freiheit in Syntax und Notation einräumt. Die Idee, Lyrik in Perl zu schreiben, lag daher offensichtlich näher, als dasselbe in Algol zu versuchen. 1990, drei Jahre nach der Erfindung der Programmsprache, schrieb Larry Wall selbst das erste, als Programm allerdings dysfunktionale Perl-Gedicht.[25] Kurz darauf wurde das Schreiben von Perl Poems, ähnlich dem Verfassen von Haikus und Limericks, zu einer populären Schreibform unter Perl-Programmierern. Sharon Hopkins' Aufsatz *Camels and Needles* resümiert schon 1991 einen Formenkanon der Perl-Lyrik: Nichtfunktionale Perl-Gedichte, zum Teil strophenartig, zum Teil in freien Versen notiert, als Programme funktionale Perl-Gedichte, „keyword poems“, die ein Perl-Befehlswort zum Ausgangspunkt haben, humoristische „wordsalad poems“, die Befehlswörter travestieren.[26] Im Liebesgedicht *down.pl* gewinnt die Strophe

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

durch den Programmcode poetische Dichte, zumindest für solche Leser, die auch die Perl-Syntax lesen können: Das Befehlswort „sub“ eröffnet ein Unterprogramm und das Dollarzeichen eine Textspeichervariable, deren Gültigkeit durch das Präfix „my“ auf das Unterprogramm beschränkt wird. Nicht nur spricht hier also ein „Ich“ zur Geliebten, sondern die Gedichtzeilen sprechen auch in der ersten Person zu ihrer Prozessierung; ihre „Sinne“ sind ein Gedächtnis, das sich im Subprogramm unterwirft. Ausführbare Perl Poems (wie z.B. das im Jahr 2000 für einen Perl Poetry-Wettbewerb eingereichte Gedicht *self delusion*) sind sogar dreifach lesbar; erstens als Gedicht in natürlicher Sprache, zweitens als Sequenz von Maschinenbefehlen, und drittens, sobald es ausgeführt wird, als neues Gedicht in natürlicher Sprache.

Wie keine andere Form der Netzkunst und Netzdichtung ist die Programmiersprachen-Dichtung strukturell mit dem Computer verknüpft und an ihn gekoppelt. Einen konzeptuellen Schritt über sie hinaus unternimmt Netzdichtung, die ihre Sprache an Computerprogrammiersprachen geschult hat, ohne in strikter Programmiersyntax geschrieben zu sein. Die australische Netzlyrikerin mez (Mary Anne Breeze) dichtet in einer selbsterfundenen Kunstsprache „mezangelle“, in der sich ASCII Art und Pseudo-Programmcode vermischen. Ein kürzerer Beispieltext, der im Juni 2000 als E-Mail-„Spam Art“ unter anderem über die Netzkunst-Verteilerlisten *7-11*, *Nettime*, *recode* und *wryting* verschickt wurde:

```
::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
```

[25] print STDOUT q
Just another Perl hacker,
unless $spring
nach [Hop91], S.6

[26] [Hop91], S.2-4

```
the first try[mester].


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

Ähnlich den *portmanteau words* von Lewis Carroll und James Joyce verschachteln sich die Wörter der mezangelle doppel- und mehrdeutig ineinander. Die Klammer-Notation ist Programmiersprachen und der Boolschen Algebra entlehnt, wo sie mehrere Zeichen gleichzeitig referenzieren, also Vieldeutigkeit beschreiben. Diese Polysemie ist, wie in vielen Texten von mez, auch eine der Geschlechter: „fe[male]tus“ liest sich simultan als „Fötus“, „weiblich“ und „männlich“. Andere Wörter wurden durch Punkte segmentiert wie Dateinamen und Verzeichnisbäume gängiger Computerbetriebssysteme: „AC.TIF.[f]ASHION]“ zum Beispiel spielt mit dem „.tif“-Namenskürzel von Graphikdateien. „Fantazee Genderator“ und „Vari.able[bodie]z“ sind durch ihre Interpunktions-Präfixe und -Suffixe, die sich an die Zitierkonventionen von E-Mail und Chats anlehnen, als Namen von Sprechern lesbar. Die Sprache greift orthographische Manierismen der Raubkopierer-Kultur auf, z.B. in den Ersetzungen von „s“ durch „z“. Die Rede ist von fünf Märchengestalten – „Prince Cessspit N Princess Pit N Cin.der.ella[fitzgeraldingz] N Rap[t]punzelle N Gr.etal]“ –, die aber nur Datenkörper („Vari.able[bodie]z“) sein könnten.

Die Mehrdeutigkeiten des Texts erschöpfen sich nicht im Sprachcode der „mezangelle“. Er provoziert auch Fragen wie: Wurden Teile von ihm durch programmierte Filter modifiziert oder durch Algorithmen generiert, und wenn ja, welche? Basiert der Text auf im Netz vorgefundenem oder mitgeschnittenem Material? Soll seine Massen-Verschickung Reaktionen provozieren, die mezangellisiert werden, um rekursiv stets neue Reaktionen und Mezangellisierungen zu provozieren?

SUB MERGE {MY $ENSES; ASCII ART, REKURSION, LYRIK IN PROGRAMMIERSPRACHEN

Zwar mögen die Textspiele der Programmierer-Netzkultur in ihren ersten Formen der ASCII Art-Teddybären und Perl-Erbauungsgedichte naiv anmuten. Mit den neueren, konzeptkünstlerischen Umcodierungen dieser Spiele aber hat nicht nur eine sentimentalisch-manieristische Periode der Netzpoesie begonnen, sondern es zeigt sich auch, daß ihr Potential unterschätzt wurde. Gleich ob in naiver oder disruptiver Form, schreiben ASCII Art, Rekursion und Programmiersprachen-Lyrik eine Poesie, die im Gegensatz zur „Hyperfiction“- und „Multimedia“-Netzliteratur den Computer und das Internet eben nicht simpel als Befreiungsinstrumente oder Expansionen anderer Medien ansieht. Stattdessen ist ihr technischer Aufwand minimal und sind potentielle Störungen, Inkompatibilitäten und Fehlcodierungen Teil ihrer Poetik und Ästhetik. Indem sie den Computer vor allem als selbstbezüglichen Generator kontingenter Zeichen versteht, als System von Kontrolle und Absturz, ist ihr Verständnis digitaler Zeichen skeptischer und auch politischer als das einer Literatur, die mit Hypertext-Verweisen und digitaler Vernetzung immer noch eine ästhetisch-gesellschaftliche Utopie verbindet.
\bibliographystyle{alphadin} \bibliography{submergemysenses}
\end{document}

LITERATUR

[Bar68] BARTH, John: *Lost in the Funhouse*. New York, London, Toronto, Sydney, Auckland : Doubleday, 1988 (1968) (Anchor Books)

[Bau99] BAUMGÄRTEL, Tilman: *net.art*. Nürnberg : Verlag für moderne Kunst, 1999

[BK93] BOEHNCKE, Heiner (Hrsg.) ; KUHNE, Bernd (Hrsg.): *Anstiftung zur Poesie : Oulipo – Theorie und Praxis der Werkstatt für potentielle Literatur*. Bremen : Manholt, 1993

[Djo99] DJORDJEVIC, Valentina: Textverarbeiter und Screendesigner. In: MODERNE KUNS NÜRNBERG, Institut für (Hrsg.): *netz.kunst. Jahrbuch 98-99*. Nürnberg : Verlag für Moderne Kunst, 1999, S. 18–21

[Döh98] DÖHL, Reinhard: Von der ZUSE Z 22 zum WWW. Helmut Kreuzer zum 70sten. 1998. – `http://www.stuttgart.de/stadtbuecherei/zuse/zuse_www.htm`

[Fou77] FOURNEL, Paul: Computer und Schriftsteller. In: *Anstiftung zur Poesie* (siehe [BK93]), S. 67–73

[Gys78] GYSIN, Brion: Permutation poems. In: *The Third Mind*. New York : Viking, 1978

[Hof79] HOFSTADTER, Douglas R.: *Gödel, Escher, Bach*. 12. Stuttgart : Klett-Cotta, 1989 (1979)

[Hop91] HOPKINS, Sharon: Camels and Needles : Computer Poetry Meets the Perl Programming Language. 1991. – `http://www.wall.org/~sharon/plpaper.ps`

[Les00] LESSIG, Lawrence: *Code and Other Laws of Cyberspace*. New York : Basic Books, 2000

[MB98] MATHEWS, Harry (Hrsg.) ; BROTCHIE, Alastair (Hrsg.): *Oulipo Compendium*. London : Atlas Press, 1998

[Mol71] MOLES, Abraham A.: *Kunst und Computer*. Köln : DuMont, 1973 (1971)

[Qui62] QUINE, W.V.: Paradox. In: BARTLETT, Steven J. (Hrsg.): *Reflexivity. A Source-Book in Self-Reference*. Amsterdam, London, New York, Tokyo : North-Holland, 1992 (1962), S. 21–35

[rayoJ] RAYMOND, Eric S. (Hrsg.): The Jargon File. o.J.. – `http://www.catb.org/jargon/`

[SB99] SUTER, Beat (Hrsg.) ; BÖHLER, Michael (Hrsg.): *Hyperfiction*. Basel, Frankfurt/M. : Stroemfeld, 1999 (Nexus 50)

[Tur84] TURKLE, Sherry: *Die Wunschmaschine*. Reinbek : Rowohlt, 1986 (1984)

[Tzu64] TZU, Chuang: *Basic Writings*. New York : Columbia University Press, 1964

[Var81] VARELA, Francisco: Der kreative Zirkel. In: WATZLAWICK, Paul (Hrsg.): *Die erfundene Wirklichkeit*. München, Zürich : Piper, 1994 (1981) (Serie Piper), S. 294–309

[Wag71] WAGENKNECHT, Christian: Proteus und Permutation. In: *Text und Kritik* 30 (1971), S. 1–11

[WD00] WEIBEL, Peter (Hrsg.) ; DRUCKREY, Timothy (Hrsg.): *net condition*. Cambridge, Mass. : MIT Press, 2000

[Zim97] ZIMMER, Dieter E.: Schone Gruse aus dem Netz. In: *Deutsch und anders*. Reinbek : Rowohlt, 1997, S. 272–292