A Digital Talk Disk Book

Scientific Explorations of Computer Game History in the Digital Humanities

Inhaltsverzeichnis

A Digital Talk Disk Book

Scientific Explorations of Computer Game History in the Digital Humanities

STEFAN HÖLTGEN:

Ritter Eduard, Ritter Kunibert und die Frage, wer den ersten Stein geworfen hat. Zwei BASIC-Spiele für Atari Homecomputer 'von unten' betrachtet

Empfohlenen Soundtrack auswählen:
Shodan2k
zurück

RITTER EDVARD, RITTER KUNIBERT UND DIE FRAGE, WER DEN ERSTEN STEIN GEWORFEN HAT

Zwei BASIC-Spiele für Atari-Homecomputer ’von unten’ betrachtet

STEFAN HÖLTGEN

Der folgende Beitrag widmet sich der Frage, wie zwei Computerspiele miteinander vergleichbar sind und wo die Unterschiede zu suchen wären, wenn sich diese Spiele oberflächlich ähneln – das heißt: demselben Spielgenre angehören, ähnliche Spielmechaniken und audiovisuelle Ausgabe besitzen, sich ähnlich steuern lassen, kurzum: Adaptionen ein und desselben Spielthemas darstellen. Der Vorwurf,

dass ein jüngeres Spiel lediglich die "Kopie" eines älteren sei, ist in der Geschichte der Video- und Computerspiele nicht neu. Sogar einer der Gründungsmy-then der Gattung beschäftigte sich mit einem solchen Vorwurf, der schließlich sogar vor Gericht ausgetragen wurde: War das Spiel "Pong", das die Firma Atari 1972 als Arcade-Automat publizierte, tatsächlich eine 'Kopie' des von Ralph Baer für die Konsole Magnavox Odyssey kurz zuvor entwickelten "Table Tennis"-Spiels? Die Ähnlichkeit beider Spiele auf ihren Oberflächen war frappierend — und dennoch entschied ein Richter, dass sich die "Unterflächen" zu stark voneinander unterschieden, als dass der Vorwurf der Kopie gerechtfertigt wäre (vgl. Baer, 2005:126–132 sowie Pias, 2003).

Das Begriffspaar "Oberfläche" versus "Unterfläche" hat der Informatiker Frieder Nake eingeführt, um die grund-

VICE (C64SC)

File Edit Snapshot Preferences Help

sätzliche Differenz zu markieren, die Computerprozesse begleitet:

> Das Bild als digitales Bild ist zuvorderst algorithmisch geworden: Es besitzt nun auch eine unterflächliche Innerlichkeit bzw. ist Oberfläche und Unterfläche zugleich. Beide — das ist entscheidend — sind objektiv vorhanden. Die Oberfläche des digitalen Bildes ist sichtbar, während die Unterfläche bearbeitbar ist. Die Oberfläche besteht für den Benutzer, die Unterfläche für den Prozessor (mit Programm). Zur Unterfläche gehört das und nur das, was als Datenstruktur und Algorithmus vorhanden ist.
>
> (Nake 2005:47)

Diese Grenze zwischen der Oberfläche und der Unterfläche digitaler Medien ist es, an der sich zwei Kulturen scheiden und die gänzliche unterschiedliche Lese- und

s003/001-025

warp: pause:

771% cpu
386.3 fps

CRT
Mixer

Tape: 000

Joysticks:

8 0:33.0

Schreibkompetenzen (‘literacies’) erfordert: die der Entwickler:innen und die der Nutzer:innen.(1) In einer kurzen Epoche(2) war diese Grenze durchlässig – diejenigen, die die Spiele spielten, waren auch oft diejenigen, die sie programmierten. Wenn hierbei eine fremde Programmidee adaptiert werden sollte, musste sich der Adaptionsprozess sich nicht bloß an der Emulation der Oberflächen orientieren, sondern es war auch prinzipiell möglich, die Unterfläche der Vorlage zu lesen und zu adaptieren, da es sich dabei um symbolischen Programmcode handelte, der in den meisten Fällen entweder veröffentlicht vorlag oder prinzipiell leicht eingesehen werden konnte. Eine solche Adaptionsleitung an einem konkreten Fall nachzuvollziehen, ist das Thema meines Beitrags, für den ich mir einen beispielhaften Fall herausgesucht habe, in dem folgender Vorwurf des Plagiates erhoben wurde:

VICE (C64SC)

File Edit Snapshot Preferences Help

> […] Daß KE-SOFT nicht gerade zimperlich mit Originalideen ist, sah man auch am Spiel GMORK-ATTACK, auf einem der letzten ZONGs. Das Spiel ist eine hemmungslose, und noch dazu schlechte Kopie des Original-BASIC-Spiels von Thomas Fischermann, erschienen 1985 im HomeComputer! 'AdR: DAS "ORIGINAL-SPIEL" IST UNS NICHT BEKANNT! […]'
>
> (Schönfeld 1991:7)

Das "Original-BASIC-Spiel[e] von Thomas Fischermann", das hier angesprochen wird, heißt "The Castle" und wurde als Programmlisting in der Zeitschrift "HC – Mein Home-Computer" (Fischermann 1985) veröffentlicht. Solche Veröffentlichungen dienten dazu, dass Leser:innen der jeweiligen Zeitschrift solche Programme in ihre eigenen Computer eingeben (abtippen) konnten, um dann die Software (in diesem Fall das Spiel "The Castle") nutzen zu können. Im Prinzip waren sol-

s005/001-025

warp: pause:

756% cpu
379.0 fps

CRT
Mixer

Tape: 000

Joysticks:

8 0:33.0

VICE (C64SC)

File Edit Snapshot Preferences Help

che Programmlistings daher Open Source
und konnten von den Abtipper:innen auch
nach eigenem Gusto modifiziert oder auf
fremde Systeme portiert werden.

Was aber ist an dem Vorwurf Schönfelders
an die KE-Redaktion (nicht namentlich
genannt: an Yoda Zhang alias Kemal
Ezcan, der "Gmork Attack" programmiert
hatte) dran, dass es sich dabei um eine
"schlechte Kopie" handele? Beide Spiele
ähneln einander auf ihren Oberflächen,
zeigen dasselbe Spielprinzip, erzählen
ähnliche Hintergrundgeschichten und lau-
fen auf derselben Plattform (8-Bit-
Atari-Computer). Mein nachfolgender Ver-
such, diese Frage zu beantworten, soll
aber nicht der Wahrheitsfindung dienen,
sondern ein Instrumentarium entwickeln,
nachdem Sourcecodes von BASIC-Programmen
auf Ähnlichkeiten und Unterschiede un-
tersucht werden können. Hierbei geht es
um individuelle Programmierstile, aber

s006/001-025

warp: pause:

758% cpu
380.1 fps

CRT
Mixer

Tape: 000

Joysticks:

8 0:33.0

auch um unterschiedliche Programmierkul-
turen.(3)

Untersuchungen dieser Art stellen den
zentralen Diskursen der Video- und Com-
puterspiel-Historiografie eine alterna-
tive Perspektive zur Seite, die die ar-
kane Subkultur der Spiele-Entwickler:in-
nen und die Vielfalt der verwendeten
Technologien und Stile ins Bewusstsein
der Forschung rücken. Dies führt der
historischen Forschung nicht nur neue
Quellen(4) und zusätzliche technische
Argumente für bestehende Diskurse zu,
sondern öffnet auch ein neues Diskurs-
feld: das der Technikgeschichte der Pro-
grammierkulturen und ihrer Artefakte.
Neben der Vergleichbarkeit von Programm-
codes lassen sich textkritische Fragen
zu Rezeption, Edition und eine Wissens-
geschichte der (Spiele)Programmierung
mit Blick auf die Codes beantworten.

Zunächst sollen beide Spiele mit Blick auf ihre (oberflächlichen) Ähnlichkeiten vorgestellt werden: Beide handeln von Rittern (bei Fischermann heißt er Eduard, bei Ezcan Kunibert), die ihre Burg vor Angreifern verteidigen. Letztere steigen an Leitern die Burgmauer empor. Sobald einer von ihnen oben angekommen ist, ist das Spiel verloren. Zur Abwehr können die Ritter Steine, die an den Bildrändern gelagert sind, herbeiholen und hinabwerfen. Das Motiv der eine Burgmauer erklimmenden Feinde, die durch Hinabwerfen (oder Hinabgießen) von Gegenständen von der Eroberung der Burg abgehalten werden sollen, ist aus zahlreichen Abenteuererzählungen bekannt. Es findet sich bereits in früheren professionellen Computerspielen(5) und ist leicht auch für Anfäger:innen in BASIC implementierbar.

1. Die Autoren ...

Um dem Vergleich beider Programme einen diskursiven Hintergrund zu geben, sollten auch die beiden Autoren und ihre Programmierer-Karrieren, das Publikationsumfeld ihrer Programme sowie die Hissensvernetzung der damaligen Zeit angesprochen werden. Die Programme wurden in unterschiedlichen Umfeldern und in verschiedenen Medien publiziert – beide entstanden für 8-Bit-Homecomputer von Atari. Es kann davon ausgegangen werden, dass sich die deutschsprachigen Leserschaften beider Publikationen deutlich überschnitten, da es für diese Computerplattform (anders als beispielsweise für den C64) im genannten Zeitraum nur wenige spezifische deutschsprachige Periodika gab.

Thomas Fischermann hat sein Spiel "The

Castle" im Jahr 1984 als 16-jähriger
Schüler auf seinem Atari 800 XL in She-
phardson BASIC (dem in seinem Computer
eingebauten Dialekt) programmiert. Pub-
liziert wurde es 1985 in der Juli-Ausga-
be der deutschsprachigen Computerzeit-
schrift "HC – Mein Home-Computer"(6) in
der Rubrik "Computer-Praxis", welche
BASIC-Listings für unterschiedliche Ho-
mecomputer-Plattformen enthält. Seinem
Programm ist ein Prätext vorgeschaltet,
in dem die Hintergrundgeschichte des
Spiels erzählt sowie einige Informatio-
nen zum Gameplay, der Spielmechanik und
dem Programmaufbau gegeben werden.

Neben BASIC-Spielen entwickelte
Fischermann auch Tools in Maschinenspra-
che, wie "Happy-Mon", "Bernoullis
Dream", einen "Latein-Vokabeltrainer",
Druckertreiber, Makros für eigene und
fremde Programmentwicklung und mit "A
Musician's Dream" "ein umfangreiches

Sound-Editing-Tool für die komplette
Ausnutzung der Chips zum Zweck der Ton-
erzeugung, mit einer eigenen Interrupt-
Routine für den Einbau der fertigen So-
unds und Musikstücke in Spiele." (7) Die
Programmierung in BASIC und später Ma-
schinensprache hat er sich selbst mit
Hilfe von Programmier-Lehrbüchern, aber
auch durch Lesen fremder Sourcecodes
beigebracht:

Die Kurse in den Zeitschriften habe
ich nie durchgearbeitet, ich hatte
damals mit Büchern wie 'De Re Atari'
gearbeitet, den offiziellen Manuals
und viel Trial und Error, allerdings
auch viele Listings abgetippt und da-
bei sicher en passant viel gelernt.
Aus den Zeitschriften habe ich mir
aber immer wieder Anregungen geholt,
sowohl für innovative einzelne Pro-
grammroutinen (Eingabefilter ....) wie
auch für Tipps und Tricks (ideenrei-

che Ansteuerung der Prozessoren wie POKEY...)(8)

Thomas Fischermann hat zwischen 1984 und 1987 zahlreiche Programme für Atari-Homecomputer programmiert und in verschiedenen deutschsprachigen Zeitschriften — darunter mehrere in der "HC" — veröffentlicht. Nach 1987 hat er die Computerprogrammierung als Hobby aufgegeben. Er arbeitet heute als Redakteur bei der Wochenzeitung "Die Zeit".

Yoda Zhang, alias Kemal Ezcan, ist seit vier Jahrzehnten eine bekannte Persönlichkeit in der Atari-Szene. Er hat mit "ZONG" 1989 eine der ersten deutschsprachigen Zeitschriften für Atari-Homecomputer herausgegeben, in der auch BASIC-Listings abgedruckt wurden und die BASIC-Programme auf Heftdiskette anbot.(9) Ezcan hat von Mitte der 1980er Jahre bis in die Gegenwart zahlreiche

Spiele entwickelt und etliche Programm-
listings publiziert. Er verwaltet ein
Archiv mit seinen Arbeiten auf seiner
Homepage(10) und publiziert auch heute
noch dazu (Zhang 2011). Populär wurde er
vor allem als Sound-Programmierer. In
dieser Hinsicht erinnert sich auch
Thomas Fischermann an ihn: "Damals waren
in der Listing-Szene große Vorbilder un-
terwegs wie zum Beispiel Kemal Ezcan,
der Sounds mit einfachen Basic-Program-
men erzeugte, die er als Listings veröf-
fentlichte, und die gute Inspirationen
für eine Erweiterung der darstellbaren
Klangvielfalt abgaben."(11) Hier zeigt
sich bereits, dass eine eventuelle Re-
zeption nicht notwendig in nur eine
Richtung verlaufen musste, wenn beide
Autor:innen Zeitgenoss:innen waren ....

Ezcans Spiel "Gmork Attack" ist ein im
Dialekt TurboBASIC XL(12) erstelltes
Programm, das 1990 auf der Heftdiskette

zur Ausgabe 6 der Zeitschrift "ZONG" erschienen ist. Es hat 96 BASIC-Zeilen Umfang.(13) Im Heft befindet sich lediglich die Spielbeschreibung mit Hinweisen zur Steuerung sowie eine Aufforderung an die Leser:innen:

> Noch ein Hinweis […]: Diese drei Programme sind recht einfach programmiert. Es dürfte für viele also kein Problem sein, sie zu verbessern, das Spiel zu erweitern oder zusätzliche Level o.Ä. einzubauen. Wie immer sind wir gerne bereit, verbesserte Versionen (gegen Honorar) zu veröffentlichen. [A]lso: Ran an die Tasten!
>
> (Ezcan 1990:17)

Hier wird nicht nur der (oben bereits angedeutete) Open-Source-Charakter der Software deutlich, sondern auch die Möglichkeit angesprochen mit selbst geschriebenen Programmen Geld zu verdie-

nen. Für Schüler:innen stellten solche Einnahmen (von bis zu 2000 DM) eine beachtliche Summe dar, die nicht zuletzt auch eine Auszeichnung für die eigene Leistung darstellte, Ansporn zu weiteren Publikationen war und zu größerer Bekanntheit in der Community führte.

Dass hier explizit zur Modifikation des Programms aufgerufen wurde, stellt durchaus keine Besonderheit dar, sondern folgt einer damals bereits Jahrzehnte alten Praxis in Hacker-Kulturen (vgl. Levy 1984:31f.): Neben der ästhetischen Idee, die das laufende Programm zu Gesicht/Gehör bringt, existiert auch eine Ästhetik des Programmcodes, die sich im Programmierstil und den Programmfunktionen ausdrückt. Diese steht mit Veröffentlichung des Codes regelrecht zur Disposition: Wer es besser macht, etwas sinnvolles hinzufügen kann oder es für eine andere Plattform portiert, ist ex-

plizit oder implizit dazu aufgerufen es zu tun. Ezcans Adaption des Themas, das sich auch in Fischermanns Programm zeigt, steht ebenso in dieser Tradition, wie Fischermanns Version vielleicht selbst die Umarbeitung einer fremden Idee darstellt: "Ich vermute, dass 'The Castle' selber damals auch ein Knock-Off von irgendwas anderem war und bin mir gar nicht sicher, ob es noch billiger geht?"(14)

2. .... und ihre Programme

Der Vergleich zweier Computerprogramme auf ihrer Codeebene muss unterschiedliche Aspekte berücksichtigen. Dass im vorliegenden Fall beide Programme für Atari-8-Bit-Heimcomputer entwickelt wurden, erleichtert die Gegenüberstellung, weil die audio-visuellen Effekte und die Eingabetechnologien auf dieselbe Plattform bezogen sind.(15) Auch, dass in

beiden Programmen BASIC als Programmier-
sprache diente, erleichtert es Vergleiche anzustellen – und das, obwohl Ezcans Programm nicht das im Atari eingebaute BASIC, sondern den Dialekt TurboBASIC XL verwendet.

Vergleichsparameter lassen sich auf der Ebene des Programmaufbaus, der Anzahl und Art der verwendeten Befehle und Funktionen, der Technologie der Grafik- und Sound-Programmierung(16) sowie nicht zuletzt in den aus den Programmroutinen herauslesbaren Kenntnissen der Maschinenfunktionen ziehen. Erkenntnisse über den Entstehungsprozess der Programme lassen sich überdies aus der Zeilennummerierung ablesen sowie aus Rudimenten (im Code verbliebenen Programmelementen, die keine erkennbare Funktion erfüllen). Die Variablennamen liefern überdies mögliche Hinweise über die Herkunft verwendeter Routinen.

Zeitweise geben die Autor:innen selbst Hinweise zu Programmfunktionen, zum Aufbau und den Variablen. Diese finden sich entweder in Programm-Anmerkungen (REM) oder den Begleittexten – so auch bei Fischermanns Programm, wo sich im Paratext zum Programmlisting (Fischermann 1985: 82f.) sowohl Informationen zur "Programmstruktur" als auch eine "Variablenliste" finden. In der Spielbeschreibung gibt es zudem einen markanten Hinweis auf einem 'Programmierfehler':

> Beim Abtippen muß darauf geachtet werden, daß PLR$ die erste Variable ist, die eingegeben wird, sonst läuft nämlich gar nichts an dem Spiel korrekt. Am besten ist es, wenn man vor dem Abtippen einen Kaltstart durchführt und dann eingibt: PRINT PLR$. Es erscheint ein Error; nun kann das Programm eingegeben werden.
> (Fischermann 1985:82)(17)

s018/001-025

Darüberhinaus enthält "The Castle"
strukturierende Anmerkungen zur Trennung
der Programmblöcke (BASIC-Zeile/n [im
Folgenden: BZz.] 10, 80, 130, 220, 260,
400, 780, 830, 940, 1020, 1200, 1350,
1430, 1540, 1640, 1720). BZz. 1140 und
1150 enthalten REM-Anmerkungen, welche
die ATASCII-Zeichen,(18) die eingegeben
werden sollen, beschreiben. Der Hefttext
von "Gmork Attack" enthält zwar keine
technischen Angaben zum Programm jedoch
das Listing selbst: Neben den REM-Zei-
len, zu Beginn des Programms (BZz. 1–7,
mit Angaben zur Autorschaft und den Kon-
taktdaten) finden sich überschriftenar-
tige Kommentare, die die Programmblöcke
semantisch voneinander trennen (BZz. 50,
150, 1000, 1100, 1200, 5000).

Solche strukturierenden REM-Kommentare
helfen dem/der Abtipper:in beim Ver-
ständnis des Codes und bei der späteren
Fehlersuche. Fehler entstehen entweder

beim Abtippen des Programmlistings oder sind sogar manchmal noch im Code verblieben und werden dann ggf. in späteren Ausgaben der Zeitschrift in Errata und Leserbriefen moniert/korrigiert. Größere BASIC-Programme, wie sie die beiden Spiele darstellen, werden zumeist auf diese Weise modular programmiert, wobei das Hauptprogramm (die "main loop") dann die einzelnen Unterprogramme in Abhängigkeit von Programmereignissen (Eingaben, Kollisionsabfragen, Timer-Ereignisse) ansteuert. Eine solche Strukturierung verfolgt auch den Zweck, das BASIC-Programm zu beschleunigen, indem bestimmte Routinen nur fallabhängig aufgerufen werden.

Fischermanns Programm umfasst 183 BASIC-Zeilen,(19) die keine auffälligen Abweichungen in der Zeilennummerierung(20) oder Rudimente im Code enthalten, welche Rückschlüsse auf nachträgliche Korrektu-

ren zuließen, während sich solche in Ez-
cans Programm an mehreren Stellen fin-
den. Vor allem ist hier auffällig, dass
es nur einen Befehl aus dem TurboBASIC-
XL-Dialekt im Programm gibt, der so
nicht im Shepardson BASIC enthalten ist:
In BZ 155 findet sich PAUSE 3 und in BZ
1230 PAUSE 1 zur Verzögerung des Stein-
falls um jeweils 1/50 Sekunde (vgl.
Specht 1990:72). Im Umfeld dieses Be-
fehls findet sich auch das einzige Merk-
mal strukturierter Programmierung des
Listings: Die BZz. 1120 sowie 1230-1250
sind eingerückt, was sie als Unterele-
mente einer FOR-Schleife in BZ 1110 und
1220 markiert. Während das Shepardson
BASIC solche Strukturierungen ignoriert
(und bei Eingabe von Leerzeichen oder
Tabulatoren diese sogar wieder löscht),
fügt TurboBASIC-XL diese stets hinzu,
wenn Schleifen über mehrere Zeilen ange-
legt werden. Der Rest von Ezcans Pro-
gramm ist allerdings mit so wenigen

strukturellen Elementen programmiert, dass dieses Konstrukt nur an zwei Stellen im Listing erscheint und zusammen mit dem PAUSE-Befehl der einzige Hinweis auf den verwendeten Dialekt ist. (Ein Indikator für den niedrigeren Grad an Strukturierung ist auch die durchschnittlich höhere Zahl von Anweisungen pro Zeile.)

Anders als in Shepardson BASIC existiert im von Ezcan verwendeten Dialekt ein RENUM-Befehl, den er jedoch offenbar nicht genutzt hat, denn es zeigen sich an verschiedenen Stellen von dem sonst durchgängigen Nummerierungsschema in 10er-Abständen abweichende Zeilennummern: BZz. 45, 65, 155, 195, 196, 1035, 1262, 1265 und 5115. Schon im Handbuch des ersten BASIC-Dialekts wird empfohlen, BASIC-Programme zur besseren Wartbarkeit mit 5- oder 10-zeiligen Zeilenabständen aufzubauen, denn:

Notice that being able to insert a line requires that the original line numbers not be consecutive numbers. For this reason, most users will start out using line numbers that are multiples of five or ten, but that is up to them.

(DCCC 1964:22)

Zwei Beispiele aus Ezcans Programm herausgegriffen zeigen, welche editionsphilologischen Annahmen diese unregelmäßigen Zeilennummern ermöglichen.

Die Variablen-Zuweisung EASY=1 (BZ 65) ist eine offenbar nachträglich eingefügte Möglichkeit, den Schwierigkeitsgrad des Spiels einzustellen. Durch Setzen dieser Variable auf 1 werden die Ergebnisse der (wohl ebenfalls nachträglich angehängten) IF-Abfragen in den BZz. 90 und 1210 "wahr" und die Variable STEIN erhält den Wert 35. STEIN beschreibt,

was die Spielfigur in Händen hält, nachdem sie einen Stein die Mauer hinabgeworfen hat. Im schwierigen Modus sind ihre Hände danach leer und sie muss erst zum Steinhaufen am Bildrand gehen, um einen neuen Stein zu holen. Inzwischen können die Angreifer die Leitern weiter hinaufrücken. Durch Setzen auf EASY=1 hat die Spielfigur aber sofort einen neuen Stein in der Hand, spart damit Zeit, und kann die Angriffe früher abwehren. Offenbar war das Spiel schwer zu spielen, weshalb Ezcan die Variable EASY nachträglich eingeführt hat.

Das zweite Beispiel findet sich in BASIC-Zeile 1262, die folgende Statements enthält:

```
IF P/100=INT(P/100) THEN
WAR5=WAR5+0.1*(WAR5«1):SCRLF=SCRLF-(SCR-
LF»4):CTR=1
```

```
Damit wird der Fall geprüft, ob der/die
Spieler:in einen Score hat, der ein
Vielfaches von 100 ist. In diesem Fall
wird der Schwierigkeitsgrad des Spiels
erhöht, indem die Variable für die zeit-
liche HA[h]RScheinlichkeit, dass ein
Gegner erscheint und eine Leiter er-
klimmt, von 0.5 (Startwert) sukzessive
um 0.1 erhöht wird (bis zu einer Wahr-
scheinlichkeit von 100 Prozent, was
WAR5=1 entspricht). Diente die EASY-Va-
riable zur Spielerleichterung, so wurde
hier (ebenfalls offenbar nachträglich)
eine steigende Schwierigkeitskurve in
ein allzu gleichmäßiges und zu einfaches
Spiel eingefügt.

                          (Ende Teil 1)
```

(Teil 2)

3. Die Grafik

Schönfelders Vorwurf des Plagiats basiert wohl zur Hälfte auf der optischen Ähnlichkeit beider Spiele, die deren recht ähnliche Hintergrunderzählung illustriert und als Grundlage für ein fast identisches Gameplay dient. Die Grafiken beider Spiele bestehen aus den Elementen: Spielfigur, Gegnerfiguren, Leitern, Mauer(steinen) und Steinen als Wurfobjekte. Solche Grafiken lassen sich mit dem 8-Bit-Atari auf unterschiedliche Weise generieren.

Kemal Ezcan nutzt für "Gmork Attack" so genannte Pseudografik (vgl. Groner 1973:m41; Stanton/Pinal 1984:51-78; Chabris 1984; Lampton 1986: 102–117), bei der Elemente des im Computer eingebauten Zeichensatzes für Grafiken ge-

nutzt werden. Dies stellt er als die sinnvollste Technologie für BASIC-Spiele dar:

> Wenn man in Basic auf dem kleinen Atari ein Spiel entwickeln will, dass einigermassen gut aussieht, ist die Methode der Wahl für die Grafikdarstellung, den Zeichensatz umzudefinieren. Das heisst eigentlich nur, dass das was vorher z.B. ein 'A' war, danach anders aussieht, z.B. wie ein Monster, eine Mauer oder Obst.
>
> (Zhang 2011:41)

Der ATASCII-Zeichensatz bietet zwar bereits etliche Grafik-Zeichen; Ezcan erweitert diesen Vorrat für "Gmork Attack" jedoch noch um einige neue. Die Zeichensätze des Atari-Computers befinden sich schreibgeschützt in ROM-Bausteinen und sind dort jederzeit für die Software nutzbar. Um sie manipulieren und ändern

zu können, müssen sie zuvor ins RAM des Computers kopiert werden. Dann muss der Zeiger, der vorher auf die Adressen des Zeichensatzes im ROM verwiesen hat, auf die neue Adresse im RAM 'umgebogen' werden, so dass der Computer nun mit der Zeichensatz-Kopie arbeitet. Letzteres geschieht in BZ 70, wo durch POKE 756,PAGE die RAM-Adresse des neuen Zeichensatzes eingestellt wird.(21)

Jedes Zeichen des Zeichensatzes besteht aus einer 8x8-Pixel-Matrix. In BZz. 28000-28120 sind die Werte für die neuen Zeichen als so genannte Bitmaps (vgl. Abb.1) hinterlegt. Die ersten Zahlen der DATA-Zeilen enthalten den (AT)ASCII-Code des neu zu definierenden Zeichens, die nachfolgenden die Dezimalwerte des Bitmaps.

VICE (C64SC)

File Edit Snapshot Preferences Help

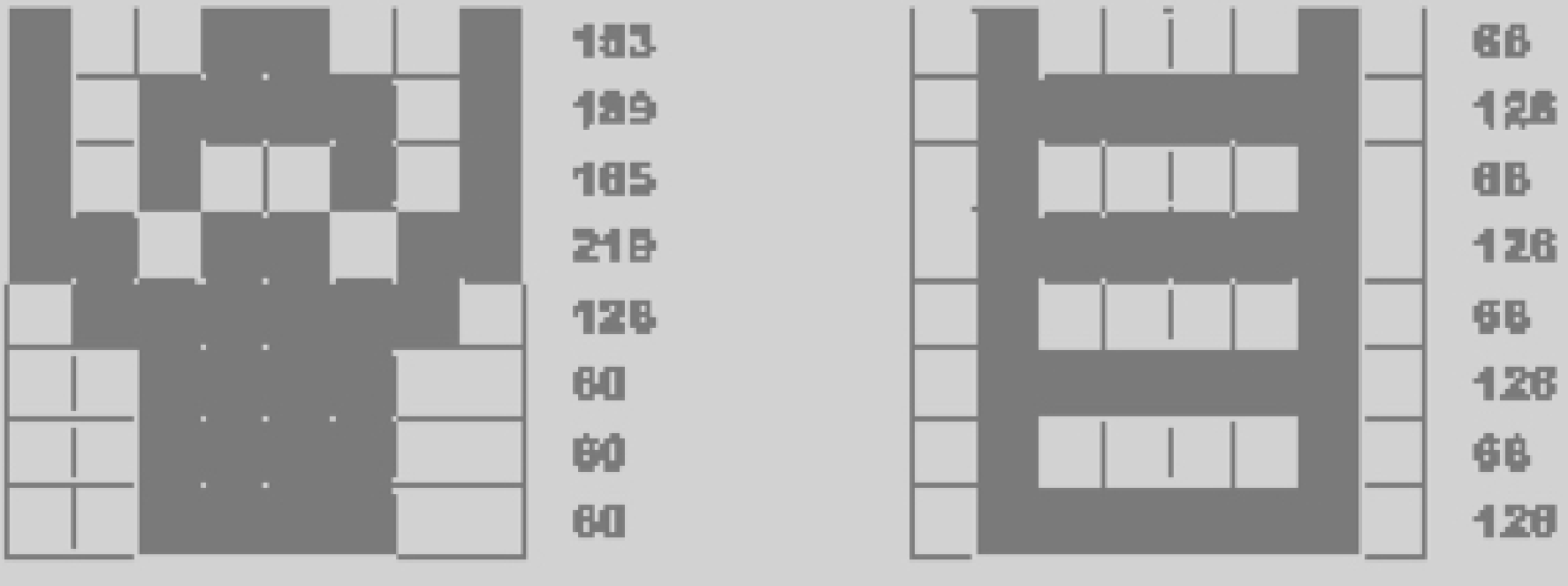

Abb.1: Bitmaps der neudefinierten Zeichen für "Gmork Attack"

Ein Nachteil der Pseudografik liegt darin, dass sich die Grafikelemente nur in 8-Pixel-Schritten horizontal und vertikal über den Bildschirm bewegen lassen. Dafür können sie bei der Programmierung

s004/001-020

warp: pause:

758% cpu
380.1 fps

CRT
Mixer

Tape: 000

Joysticks:

8 0:47.0

aber wie Zeichenketten (Strings) behandelt und verwaltet werden. So ist das Zeichen für den im Fall befindlichen Stein die ehemalige "öffnende runde Klammer" (ASCII 40) und das Zeichen für einen die Leiter hinaufsteigenden Gegner das ehemalige "Prozent"-Symbol (ASCII 37). Soll im Programm nun geprüft werden, ob der herabfallende Stein den Gegner getroffen hat, muss lediglich geprüft werden, ob sich (und % an der selben Bildschirmposition befinden, was durch den LOCATE-Befehl in BZ 1230 erreicht wird.

Dieselbe Darstellungstechnik verwendet auch Thomas Fischermann in "The Castle". Auch er kopiert zunächst die Zeichensätze ins RAM (BZz. 430, 440), um danach spezifische Zeichen daraus für seine Grafikelemente umzudefinieren. Allerdings sind bei ihm nicht alle Grafikelemente des Spiels auf diese Weise er-

stellt. Die Spielfigur selbst, Ritter
Eduard, ist eine zweifarbige Player-
Missile-Grafik.(22) Hierbei handelt es
sich um eine Technologie, mit der kom-
plexe Grafik-Elemente unterschiedlicher
Größe und Farbigkeit vom Videoprozessor
des Atari-Computers (ANTIC) erzeugt und
verwaltet werden. Der Vorteil liegt ne-
ben der pixelgenauen Positionierbarkeit
in der Möglichkeit, das Grafikobjekt un-
abhängig von der Hintergrundgrafik bewe-
gen und Kollisionen über ein Register
des Grafikchips abfragen zu können. So
ist es Fischermann möglich, Burgzinnen
auf seine Mauer zu setzen (BZ 1120),
hinter denen die Figur entlang laufen
kann; dies wäre bei der in "Gmork At-
tack" verwendeten Technik nicht möglich.
Die Verwendung von Player-Missile-Grafik
bedarf jedoch spezifischer Kenntnisse in
der Programmierung (vgl. BZ 780-930).

Auch dort, wo sich ähnliche Zeichen wie-

derholen, um ein Muster zu generieren, bietet sich der Einsatz von Pseudografik an. In beiden Spielen ist dies der Fall bei der Generierung der Mauern. Ezcan verwendet hierfür das neu definierte "$" für versetzte Ziegelsteine sowie das "'" für den angeschrägten oberen Abschluss der Steinmauer. Da sein Spiel zwei verschiedenfarbige Mauern darstellt (eine violette im Vordergrund und eine dunkelblaue im Hintergrund), werden für beide Mauerarten jeweils zwei Zeichen definiert. Beachtenswert ist dabei das Verfahren, mit dem die Mauern gezeichnet werden, weil sich hierin eine weitere Ähnlichkeit beider Spielcodes zeigt.

"Gmork Attack" bedient sich des Atari-Grafik-Modus 17(23), in dem 20 mal 20 Zeichen in fünf unterschiedlichen Farben dargestellt werden können. Dieser Modus gehört zu den Character-Grafikmodi, in denen statt Pixeln Textzeichen angezeigt

s007/001-020

werden – genau der richtige Modus also, um als Pseudografik redefinierte Spiel-elemente auf den Bildschirm zu bringen (weshalb er vergleichsweise häufig in BASIC-Spielelistings zu finden ist). Um eine Grafik zu generieren, können die Befehle PLOT und DRAWTO benutzt werden. In Pixel-Grafik-Modi werden dadurch Punkte und Linien in einem Character-Grafikmodus Buchstaben angezeigt. Die Farbe wird über den Befehl COLOR einge-stellt. In diesem Modus wird als Farb-wert der (AT)ASCII-Wert des jeweiligen Zeichens genutzt.(24) Die fünf verfügba-ren Farben sind den (AT)ASCII-Zeichen-werten zugeordnet. Die BZz. 10020-10060 zeigen, wie die Mauer auf diese Weise mit wenigen COLOR- und DRAWTO-Anweisun-gen auf den Bildschirm gebracht werden kann.

Auch "The Castle" nutzt den Grafikmodus 17. Hier findet der Bildschirmaufbau in

```
den BZ 1020-1090 statt. Fischermann ver-
wendet den DRAWTO-Befehl (BZ 1070) hier
jedoch lediglich zum Zeichnen des Fah-
nenmastes am oberen Ende der Mauer. Da-
nach werden alle Grafikelemente wie
sonst Schrift mit dem PRINT-Befehl an
den Ausgabekanal #6 (das ist der Grafik-
schirm) ausgegeben. Dies hat im Listing
einen wesentlich leichter nachvollzieh-
baren Bildschirmaufbau zur Folge: In der
FOR-Schleife in BZ 1130 werden zwölf
Zeilen mit je 17 Ausrufe-Zeichen (ASCII
33) platziert. Dabei handelt es sich um
die in den BZ 470 und 630 definierten
Pseudografikzeichen mit Mauerelementen.
Die in BZ 1140 und 1150 angegebenen
Hochkomma-Symbole bringen das Gras un-
terhalb der Mauer auf den Bildschirm;
mit den Zeichen 2, 3, 4 und 5 in BZ
1080-1110 wird die "IS"-Fahne generiert.

Vergleicht man beide Implementierungen
der Grafiken von "The Castle" und "Gmork

s009/001-020
```

Attack", so wirken sie auf den ersten Blick recht unterschiedlich durch die Verwendung von COLOR/DRAWTO bei "Gmork Attack" und PRINT bei "The Castle". Im Prinzip nutzen sie jedoch dieselbe Pseudografik-Technik, die sich bei der Nutzung des von beiden Spielen verwendeten Grafikmodus 17 anbietet. Das Mosaik-Prinzip, nach dem die Mauer aus versetzten Stein-Elementen aufgebaut ist, findet sich in beiden Programmen — wenngleich sich die Details der Mauern unterscheiden (vgl. Abb.2).

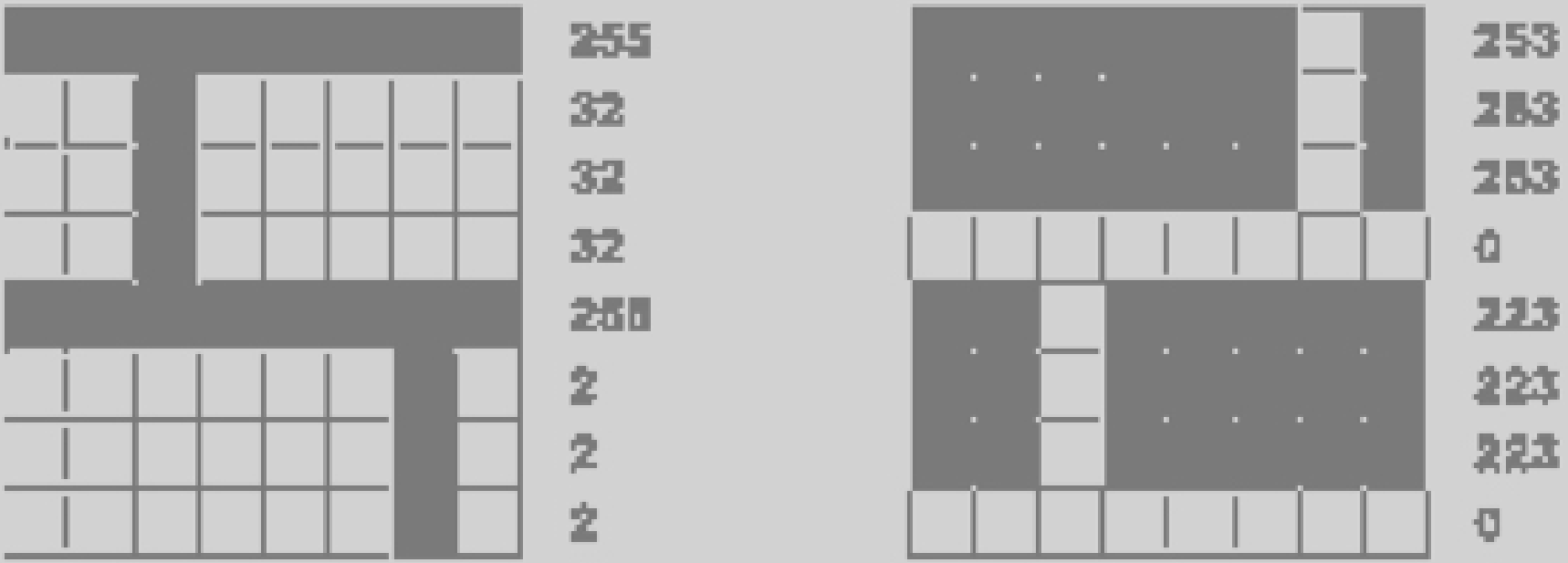

Abb.2: Berechnung der Bitmap-Grafiken

Diese Grafikelemente sind sicherlich auch für den ersten Anschein von Ähnlichkeit beider Spiele verantwortlich, bedecken sie doch den größten Teil des Bildschirms. Allerdings ist sowohl die Verwendung von Pseudografik im Grafikmo-

dus 17 als auch die Generierung von Mauern durch versetzte Stein-Elemente noch kein hinreichendes Kriterium einer "The Castle"-Adaption durch Ezcan – beides taucht in zahlreichen BASIC-Spielen auf (vgl. Höltgen 2023; S.O. 1981; Schreiber 1985:149) und wird bei Ezcan auch in anderen Spielen eingesetzt (vgl. Ezcan 1992:43/BZ 29020).(25)

4. Programmierkulturen

Welchen Schluss lässt dies nun für die Ausgangsfrage zu? Sicherlich wäre Schönfelders Vorwurf der "billigen Kopie" kaum gerechtfertigt, verstünde man ihn als Plagiatsvorwurf, bei dem Ezcan von Fischermann 'abgeschrieben' habe. Zu viele Unterschiede zeigen sich beim Vergleich der Sourcecodes; und dort, wo sich Ähnlichkeiten ergeben, verweisen diese eher auf eine gemeinsame Kodierungstradition. "Billige Kopie" ließe

sich aber auch als "weniger komplexe Ausführung" verstehen, bei der zwei neue Varianten des (Sub)Genres 'Castle Defend'-Spiel entstanden sind, die auf einen gemeinsamen "Urtext" (Roelli 2020:336f.) verweisen, den die Autoren (produktiv) rezipieren. Diese Rezeptionsleistung erscheint Schönfelder bei "Gmork Attack" stilistisch weniger komplex oder anspruchsvoll geraten zu sein, als bei "The Castle".

Als derartige Rezeptionen bilden sie nicht nur oberflächlich, sondern auch auf ihren Unterflächen "Familienähnlichkeiten" (Hittgenstein 1997:277f.) aus, deren Unterschiede Hinweise auf unterschiedliche Programmierkulturen geben. Hierbei steht jeder Algorithmus und jede Textgestalt in einer jeweils eigenen Tradition. Diejenigen, die an dieser Tradierung teilnehmen, bilden eine 'Coding Community', deren Teilnehmer sich

nicht notwendigerweise sozial oder geo-
grafisch nahestehen müssen. Ihr Rezepti-
ons-Netzwerk bildet sich über die ge-
meinsam genutzten Medien(kanäle) aus,
die vor allem durch Sekundärliteratur
(Zeitschriften und Bücher) und die ge-
meinsam verwendeten Plattformen eta-
bliert und stabilisiert werden.

An den Stellen, an denen sich die Arte-
fakte (auch bloß in Details) voneinander
unterscheiden, offenbaren Idiosynkrasien
das jeweils unterschiedliche Computer-
wissen und die unterschiedlichen Formen
und Niveaus einer (Auto-)Didaktik des
Programmierens oder es zeigen sich in
den Verzweigungen und Knoten vielleicht
Hinweise zu anderen Coding Communities.
Dieses Netzwerk aus Historemen nachzu-
zeichnen wäre ein wichtiger Beitrag zu
den Historiographien der Computerspiele,
der Programmierung, der Genealogie von

Algorithmen und nicht zuletzt der ‘Digi-
talisierung’ der Gesellschaft.

5. Kultur des Programmierens

Dass es überhaupt so weit kommen konnte,
dass Privatleute im besagten Zeitraum zu
Programmierer:innen wurden, liegt si-
cherlich in einem ‘Willen zum Wissen’
begründet, der bereits in den materiel-
len und symbolischen Grundlagen der Pro-
grammierung kondensiert ist: den Hard-
ware-Plattformen und den Programmier-
sprachen. Erstere, hier: die Homecompu-
ter, stellen das kommerzielle Endprodukt
einer Hardware-Hacker-Bewegung dar, die
Mitte der 1950er Jahre in der ‘Eroberung
des Computers’ durch Studierende ihren
Anfang nahm und dann zu Beginn der
1970er Jahre in privaten Computerclubs
Fahrt aufnahm und die Grundlagen der
Home- und Personalcomputerindustrie bil-
dete. Die technischen Spezifikationen

der Homecomputer zeigen deutlich, dass hier Ideen der Privatnutzung bereits in Hardware gegossen wurden: Grafik, Sound, Spielcontroller-Ports, Anschluss an die heimische Medieninfrastruktur (Fernseher, Kassettenrekorder) und die ausführliche beigelegte Dokumentation, die in vielen Fällen dem Genre der Programmierlehrliteratur zuzuordnen wäre.

Die Programmiersprache BASIC, die in solchen Handbüchern vorgestellt wurde, ist ebenfalls das Produkt einer Demokratisierungs- und Privatisierungsbewegung: Nicht mehr nur Fachleute aus den Natur- und Ingenieurwissenschaften sollten programmieren lernen können, sondern alle Kulturteilnehmer. Die didaktischen Paradigmen und die syntaktische Gestaltung von BASIC haben innerhalb kurzer Zeit Menschen (fast) aller Altersklassen und Bildungsschichten zu Programmierer:innen gemacht. Das Fehlen (oder besser: die

Nicht-Notwendigkeit) curricularer Pro-
grammierausbildung hat sich im Rückblick
betrachtet als großer Gewinn darge-
stellt: Der Stilreichtum der damals ent-
standenen Sourcecodes, der auch durch
die Dispositive der Hardwareplattform
und des BASIC-Dialektes forciert wurde,
zeigt die kreative Überwindung von Denk-
und System-Schranken.

Die Geschichte dieser Dispositive lässt
sich aus den Computerspielen und den
Codes herauslesen. Der Unterschied etwa,
wie beide Spiele auf die Grafikausgabe
im Modus 17 zugreifen — mal mit PRINT,
mal mit DRAWTO — verweist auf zwei frühe
Episteme der Computernutzung: PRINT, das
ja nicht "schreibe", sondern "drucke"
bedeutet, gemahnt an das Zeitalter der
Terminal-Bedienung von Computern, bei
denen die Nutzer:innen über schreibma-
schinenartige Geräte mit dem zentralen
Computer kommunizierten und alle Ausga-

ben auf Papier stattfanden. Im Grafikmodus 1 (der das Textfenster im unteren Bildschirmdrittel anzeigt), ist diese Kommunikationsschnittelle noch nachvollziehbar, wenn dort mit PRINT #6 das Ausgabefenster im oberen Bildteil adressiert wird. Nutzt man hier DRAWTO, so zeichnet man mit Buchstaben (sofern sie nicht zuvor als Grafikzeichen redefiniert wurden). Die "ASCII-Art", die ebenfalls aus der Zeit der Teletype-Terminals stammt und auf den Homecomputern noch einmal eine letzte Blüte erlebte, findet ihren Widerhall in solchem 'Malen mit Symbolen'. Das sind nur zwei Details der Programme, die bei genauerer Betrachtung noch vielfältigere Epistemologien aufrufen und so weitere Parallelgeschichten des Computings erzählen.

Auf dieser Ebene betrachtet, öffnet die Softwareanalyse, bei der Sourcecodes als Textsorte und Programmlistings als Texte

betrachtet werden, den Zugang zu einer Editionswissenschaft, die sich sprach- und literaturwissenschaftlicher Methoden bedienen kann. Allerdings bedarf es hier stets eines "close reading", das die Argumente im Detail sucht und sammelt, um sie zum Beispiel für ein Archiv des Programmierwissens zu kategorisieren. Zum:r menschlichen Leser:in muss hier notwendig auch der maschinelle Leser hinzutreten, denn mit dem Computer, der den Programmcode ausführt, fügen sich zwei weitere Archive des Wissens in den Prozess ein, die von den Programmierer:innen und Spieler:innen implizit und explizit aufgerufen werden: die Möglichkeiten und Grenzen der Maschine und ihrer Komponenten sowie die in die Programmiersprache ‘eingeschriebenen’ Wissensbestände der Sprachentwickler:innen. Zwischen beiden residiert die Programmierung und die Anwendung von Software und bringt die

jeweils unterschiedlichen Kulturen von
Nutzung und Design hervor.

(Ende Teil 2)

(Teil 3)

Endnoten

NOTE 1: Diese kulturelle Differenz lässt sich als ‘Verlängerung’ der Differenz zwischen Geistes- und Naturwissenschaften sehen, wie sie Snow (1987) Ende der 1950er Jahre konstatiert hat.

NOTE 2: Bis zur Entstehung der Hacker-Kultur in den frühen 1950er Jahren hatten User:innen keinen Zugriff auf die Programmierschnittstellen von Computern; mit der Entwicklung der grafischen Nutzeroberflächen und dem WYSIWYG-Prinzip verloren sie das Interesse an der grundsätzlichen Programmierbarkeit ihrer Computer. In der Phase dazwischen waren die User an der Frage interessiert, wie aus dem Werkzeug Computer ein Werkstück werden könnte.

NOTE 3: Dies stellt eine Vorarbeit für ein derzeit an der Universität Bonn stattfindendes Forschungsprojekt zur Technik- und Kulturgeschichte der BASIC-Programmierung dar, das perspektivisch Metriken und Methoden für eine Distant-Reading-Erforschung von BASIC-Sourcecodes in den Digital Humanities vorbereitet.

NOTE 4: Die Fülle an Quellen, die das Archiv hiervon birgt, ist derzeit kaum abzusehen. Abzusehen ist allerdings die Gefahr, dass diese Quellen nicht sach- und materialgerecht bewahrt werden. Darum ist die Auseinandersetzung, die ich hier mit solchen Quellen führe, auch als Aufruf zu verstehen, diesen Objekten der Computer(spiel)geschichte größere Aufmerksamkeit zu widmen, wenn es um Initiativen der Erfassung, Erhaltung und Erforschung von Spielen geht.

NOTE 5: Das Spielprinzip findet sich ähnliche im Arcadespiel "Avalance" (Atari 1978) oder "Kaboom" (1982, Activision), bei dem die von oben herabgeworfenen Objekte jedoch durch die unten stehende Spielfigur gefangen werden müssen.

NOTE 6: "HC – Mein Heim-Computer" erschien von November 1983 bis Oktober 1986 monatlich im Vogel-Verlag und richtete sich dediziert an Besitzer:innen von 8- und 16-Bit-Homecomputern.
RTRO.DE/DB39

NOTE 7: Zitat Thomas Fischermanns aus einer E-Mail an den Autor.

NOTE 8: Zitat Thomas Fischermanns aus einer E-Mail an den Autor.

NOTE 9: Solche Magazin-Disketten wurden insbesondere in der zweiten Hälfte der

1980er Jahre populär, da bei Leser:innen von Zeitschriften das Interesse sank, die immer längeren Programmlistings (oft bestehend aus tausenden Hexadezimal-Zahlen) per Hand abzutippen. Gegen wenig Geld boten die Redaktionen an, die Programme auf Datenträgern (Heft-Disketten oder Heft-Kassetten) an solche Leser:innen zu versenden.

NOTE 10: RTRO.DE/DB40 sowie RTRO.DE/DB41

NOTE 11: Zitat Thomas Fischermanns aus einer E-Mail an den Autor.

NOTE 12: TurboBASIC XL wurde als Erweiterung und Beschleunigung des in die Atari-Computer eingebauten Shepardson-BASIC von Frank Ostrowski 1985 in der Zeitschrift "Happy Computer" veröffentlicht, hat sich in der Atari-Szene stark verbreitet und wurde zu einem Quasi-Standard, der bis heute in Retrocompu-

ting-Szenen verwendet wird. Vgl.
RTRO.DE/DB42 sowie Ostrowski 1985.

NOTE 13: Das sind durchschnittlich 1,61
Anweisungen pro BASIC-Zeile.

NOTE 14: Zitat Thomas Fischermanns aus
einer E-Mail an den Autor.

NOTE 15: Gerade im Hinblick auf ihre I/O
unterscheiden sich die Homecomputer der
bezeichneten Epoche am deutlichsten von-
einander.

NOTE 16: Für diese habe ich im Fall von
"The Castle" bereits Vorarbeiten geleis-
tet (vgl. Höltgen 2022).

NOTE 17: Ataris Shepardson-BASIC unter-
scheidet sich von vielen anderen BASIC-
Dialekten dadurch, dass es ein halbkom-
piliertes BASIC ist: Bereits während der
Eingabe der Programmzeilen (und nicht

```
erst später beim ersten Lauf) werden die
Eingaben auf syntaktische Richtigkeit
geprüft. Bei einem Fehler erhält der/die
Abtipper:in sofort eine Fehlerausgabe,
falls ein Vertipper vorliegt. Dies
scheint im vorliegenden Fall auch dazu
zu führen, dass die Variable PLR$ Ein-
fluss auf die Programmverarbeitung beim
Abtippen und späteren Ausführen des Pro-
gramms nimmt. Hier zeigt sich einer (der
nicht wenigen — vgl. RTRO.DE/DB43 und
RTRO.DE/DB44 Programmierfehler des
BASIC-Interpreters.

NOTE 18: ATASCII ist die Atari-Erweite-
rung des ASCII-Zeichensatzstandards.
Hierin sind Grafikzeichen, invers darge-
stellte Typen und andere Sonderzeichen
enthalten. In den genannten BASIC-Zeilen
wird das ATASCI-Zeichen 167 (ein invers
dargestelltes Hochkomma) beschrieben,
das vielleicht im Listing-Abdruck nicht
gut zu erkennen gewesen wäre.


s006/001-014
```

NOTE 19: Mit dem Doppelpunkt können mehrere Anweisungen in einer BZ untergebracht werden. In "The Castle" finden sich durchschnittlich 1,31 Anweisungen pro Zeile.

NOTE 20: Es ist allerdings schon mehr als auffällig, dass der 183 Zeilen lange Programmcode in regelmäßigen 10-Zeilen-Abständen aufgebaut ist. Da das Shepardson-BASIC über keinen RENUM-Befehl verfügt (mit dem nachträgliche Änderungen der Zeilennummerierung vorgenommen werden können), bedeutet dies entweder, dass Fischermann sein Programm 'in einem Stück' funktionierend implementiert hat (was wohl ausgeschlossen werden darf) oder dass er es 'vorgeschrieben' hat und diese Vorversion für die spätere Veröffentlichung noch einmal geglättet hat. Dies wäre etwa dadurch möglich, dass der Autor das Programm in TurboBASIC XL geladen, mit RENUM formatiert und dann

wieder gespeichert hat, da dieser Dialekt voll abwärtskompatibel zum Shepardson-BASIC ist.

NOTE 21: RTRO.DE/DB45

NOTE 22: Bei anderen Heimcomputern heißen solche Grafikelemente oft "Sprites".

NOTE 23: Die Grafik-Modi 0-15 verfügen über 4-zeiliges Textfenster am unteren Bildrand. Dieses kann durch Addition von 16 zum Grafikmodusparamenter ausgeblendet werden. Beim Grafikmodus 17 (1+16) handelt es sich also um den Grafikmodus 1 ohne Textfenster (vgl. RTRO.DE/DB46).

NOTE 24: "The two high bits of each ATASCII character, that normally identify lowercase or inverse video text in Graphics 1, set the color register for the 64 character set. Decimal character numbers 0-63 use color register zero, while

those same 64 characters if given cha-
racter numbers 64-127 use color register
#1. If you are typing from the Atari
keyboard, the uppercase letters A-Z ATA-
SCII 65-90 (Internal # 33-58) are assig-
ned to color register zero, while the
lowercase numbers 97-122 (Internal # 97-
122) are signed to register #1."
RTRO.DE/DB47

NOTE 25: Die "Breakout"-Variante "Smash-
out" für den Sinclair ZX Spectrum ver-
wendet dieselbe Technik zur Zeichnung
der Mauer (vgl. Höltgen 2023).

Bibliografie

Baer, Ralph (2005): Videogames in the
Beginning, Springfield: Rolenta Press.

Chabris, Chr. (1984): Character Gra-
phics. Redefine the set anyway you choo-

se, in: ANTIC, Jg. 2, Nr. 11, S. 60–69.

DCCC (Dartmouth College Computation Center) (1964): BASIC. A Manual for BASIC, the elementary algebraic language designed for use with the Dartmouth Time Sharing System, Dartmouth: Dartmouth College.

Ezcan, Kemal (1990): Gmork Attack, in: ZONG, 06/1990, S. 17.

Ezcan, Kemal (1992): Ball Harbour, in: ZONG, 08/1992, S. 37, 42f.

Fischermann, Thomas (1985): The Castle, in: HC – Mein Home-Computer, No. 07 (Juli 1985), S. 82–85.

Genette, Gérard (1993): Palimpseste. Die Literatur auf zweiter Stufe, Frankfurt am Main: Suhrkamp.

Groner, Gabriel F. (1973): Display Terminals can help People to use Computers, in: AFIPS '73: Proceedings of the June 4-8, 1973, national computer conference and exposition, New York, 4.-8. Juni 1973, S. m39–m42.

Höltgen, Stefan (2023): Das wichtigste Bit, in: Gerastertes Wissen. Pixel in Kultur und Technik, 02.06.2023, RTRO.DE/DB48

Höltgen, Stefan (2022): "Und wenn ich diese Taste drück' ..." Sounds und ihre Programmierung in BASIC-Spielen, in: PAIDIA, 23. März 2022, RTRO.DE/DB49

Lampton, Christopher (1986): Graphics and Animation on the Atari, New York u.a.: Franklin Watts.

Levy, Steven (1984): Hackers. Heroes of the Computer Revolution, Garden City/New

York: Anchor Press/Doubleday.

Nake, Frieder (2006): Das doppelte Bild. Bildwelten des Wissens, in: Kunsthistorisches Jahrbuch für Bildkritik, Jg. 3, Nr. 2, S. 40–50.

Ostrowski, Frank: TurboBASIC XL. Listing des Monats, in: Happy Computer, 12/1985, S. 28, 81–100.

Pias, Claus (2003): Mit Computern spielen. Ping/Pong als Urszene des Computerspiels. In: Stefan Poser/Karin Zachmann (Hgg.): Homo faber ludens. Geschichten zu Wechselbeziehungen von Technik und Spiel. Frankfurt am Main: Peter Lang, S. 255–279.

Roelli, Philip (Hg.) (2020): Handbook of Stemmatology. History, Methodology, Digital Approaches. Berlin/Boston: De Gruyter.

S. O. (1981): Chicken — A Great Game. In: ANTIC, Vol. 1, No. 1 (04/1982), S. 19—21.

Schönfeld, Harald (1991): Leserbrief. In: ZONG, 01/1991, S. 6-8.

Schreiber, Linda M. (1985): Atari Fun & Games. Blue Ridge Summit: Tab Books Inc.

Snow, C. P. (1987): Die zwei Kulturen. 1959. In: Helmut Kreuzer (Hg.): Die zwei Kulturen. Literarische und naturwissenschaftliche Intelligenz. C. P. Snows These in der Diskussion. dtv, München.

Specht, Rolf A. (1990): Handbuch zum Turbo BASIC XL 1.5. Herten: ABBUC e.V.

RTRO.DE/DB50

Stanton, J./Pinal, D. (1984): Atari Graphics & Arcade Game Design. Los Angeles: Arrays.

Wittgenstein, Ludwig (1997): Philosophi-
sche Untersuchungen. In: Ders.: Werkaus-
gabe Band a: Tractatus logico-philoso-
phicus, Tagebücher 1914-1916, Philoso-
phische Untersuchungen. Frankfurt am
Main: Suhrkamp, S. 225–580.

Zhang, Yoda (alias Kemal Ezcan): GULP
SPLAT ZONG. Videospiele & Computermusik.
O. O.: Yoda Zhang.