media/rep/

Repositorium für die Medienwissenschaft

Stefan Höltgen

JUMPs durch exotische Zonen. Portale, Hyperräume und Teleportation in Computern und Computerspielen

2015

https://doi.org/10.25969/mediarep/15000

Veröffentlichungsversion / published version

Sammelbandbeitrag / collection article

Empfohlene Zitierung / Suggested Citation:
Höltgen, Stefan: JUMPs durch exotische Zonen. Portale, Hyperräume und Teleportation in Computern und Computerspielen. In: Thomas Hensel, Britta Neitzel, Rolf F. Nohr (Hg.): *»The cake is a lie!« Polyperspektivische Betrachtungen des Computerspiels am Beispiel von PORTAL*. Münster: LIT 2015, S. 107–134. DOI: https://doi.org/10.25969/mediarep/15000.

Erstmalig hier erschienen / Initial publication here:
http://nuetzliche-bilder.de/bilder/wp-content/uploads/2020/10/Hensel_Neitzel_Nohr_Portal_Onlienausgabe.pdf

Nutzungsbedingungen:
Dieser Text wird unter einer Creative Commons - Namensnennung - Nicht kommerziell - Weitergabe unter gleichen Bedingungen 3.0/ Lizenz zur Verfügung gestellt. Nähere Auskünfte zu dieser Lizenz finden Sie hier:
http://creativecommons.org/licenses/by-nc-sa/3.0/

Terms of use:
This document is made available under a creative commons - Attribution - Non Commercial - Share Alike 3.0/ License. For more information see:
http://creativecommons.org/licenses/by-nc-sa/3.0/

DFG Deutsche Forschungsgemeinschaft

Philipps Universität Marburg

Stefan Höltgen

JUMPs durch exotische Zonen Portale, Hyperräume und Teleportation in Computern und Computerspielen

Einsprung

Give me light – Give me action
At the touch of a button
Flying through hyper-space
In a computer interface
Trans-X: Living on Video

Der so genannte »Portal-Effekt« bildet den spielmechanischen Ausgangspunkt für Portal und Portal 2. Er ist jedoch wesentlich älter, genau genommen: so alt wie das digitale Computerspiel selbst. Seine Wurzeln reichen von dort aus zudem in die Literatur- und Kulturgeschichte sowie in die Geschichte der modernen Physik. All diesen Annäherungen liegt dasselbe Phänomen zugrunde und sie alle beziehen ihre Faszination aus derselben Frage: *Wo* befindet sich das Objekt, nachdem es in dem einen Portal verschwunden ist und bevor es aus dem anderen erscheint? Diese im Folgenden ›exotische Zonen‹ genannten Aufenthaltsorte möchte ich untersuchen – als ›Orte von Computerspielfiguren‹ in Spielen, die den Portal-Effekt oder ähnliche Reise-Formen simulieren.

Die Frage nach dem Ort einer Computerspielfigur erscheint auf den ersten Blick sinnlos. Der Medienwissenschaftler könnte, halb-technisch◂1 gesprochen, erwidern, dass sie falsch gestellt ist, denn eine Computerspielfigur ist ja bloß Grafik und die »besteht [...] aus Algorithmen und sonst gar nichts« (Kittler 2002, 183). Die Frage nach dem ›Wo‹ einer Computerspielfigur könnte ontologisch also erst dann einen Sinn ergeben, wenn sie überhaupt irgend*wo* sein kann, wenn es also einen Körper gibt und einen Ort, an dem er war, ist oder sein wird, an dem er *Existenz* hat.

Dem könnte man entgegnen: Wir konstruieren solch einen Ort durch unsere Wahrnehmung – mit euklidisch bestimmbaren X/Y-Koordinaten auf dem Bildschirm – und durch unsere (Inter)Aktion mit der Maschine als Signal-Menge, Speicherinhalt, Datensatz in der Materie des Mediums. Oder noch radikaler: Selbstverständlich muss die Computergrafik auch irgend*wo* gespeichert und prozessiert werden. Dieser Ort bildet ihr technisches Apriori, ihr ›Existenzial‹, wenn man so will. In den Hardwares moderner Computer(spiel)plattformen

sind eigens für diesen Zweck eingerichtete Bauteile enthalten, die auf die Gestaltbildung der Grafik/Software maßgeblichen Einfluss nehmen. In ihnen wäre der *reale Ort* der Computerspielfigur zu suchen. Die konstruktivistische (wie meines Erachtens auch die ›halb-technische‹) Betrachtungsweise bleibt für sich genommen und im Sinne Frieder Nakes zu ›oberflächlich‹, die rein technische für eine medienwissenschaftliche Antwort zu ›unterflächlich‹, denn:

»Digitale Bilder sind algorithmische Zeichen. Sie sind sichtbar (für den Menschen) und berechenbar (für die Maschine). Digitale Bilder haben eine Oberfläche und eine Unterfläche. Mit der Spanne haben es die digitalen Medien zu tun. Die digitalen Medien verlangen Algorithmik und Ästhetik, beides« (Nake o. J.).

Ich will im Folgenden versuchen, die Frage nach beidem aus einer Perspektive zwischen Ober- und Unterfläche, zwischen Diskurs und Technik, zwischen Ästhetik und Algorithmus, zwischen Kultur- und Technikgeschichte aufzuwerfen: Wo befindet sich eine Computerspielfigur – als Grafik, als Signal, als Speicherinhalt, als Element einer Genealogie und als Wissensobjekt? Beispielhaft soll diese Frage an das Computerspiel Ms. Pac-Man (Bally Midway/Namco 1982, Bally Midway/General Computer Corp.) gerichtet werden, mit einem besonderen Fokus auf den ›Portal-Effekt‹, dessen unterschiedliche Implikationen ich dazu einander gegenüberstelle.

Die Untersuchung von Ms. Pac-Man (und nicht von Portal selbst) ist der Tatsache geschuldet, dass sich unterhalb der differenten Ästhetiken, der verschiedenen Schnittstellen, der jeweils verwendeten Programmiersprachen, ja sogar der Mikroprozessorgenerationen beider Spiele dieselben Phänomene finden lassen. Ms. Pac-Man soll hier also als beispielhafte Reduktion für eine Untersuchung dienen, die auch mit Portal selbst vollzogen werden könnte. Im Gegensatz zu Portal ist der grundsätzliche Aufbau von Ms. Pac-Man, also der Code, die Architektur der Plattform und die Mikroelektronik durch jahrzehntelange Untersuchungen (insbesondere durch Retrocomputer-Enthusiasten und Hacker) offenkundig und frei zugänglich. Die grundsätzliche Ähnlichkeit zwischen beiden Spielen soll im Folgenden also als didaktische Reduktion bei der Auseinandersetzung mit der Epistemologie des ›Portal-Effektes‹ dienen und daher auf alle Programme, die diesen Effekt nutzen, übertragen werden können.

Zunächst soll dazu die Ideengeschichte exotischer Reisen durch Raum und Zeit angerissen werden, um danach ein paar Beispiele für exotische Zonen von Spielfiguren aus der Geschichte des Computerspiels zu nennen und deren narrative Einflüsse zu charakterisieren. Welchen Unterschied die Differenz zwischen

literarischer Schrift, Illustration, Diagramm und Computerspiel-Programmcode in Hinblick auf die Frage nach der *Temporalität*◂**2** macht, soll sich dann am Beispiel des Programmcodes von Ms. Pac-Man zeigen, weil dessen technische Grundlagen vergleichsweise leicht darstellbar sind. Schließlich wende ich mich den *Lokalitäten* der (Spiel-)Hardware zu, um in ihnen nach Entsprechungen zu den auf dem Bildschirm sichtbaren Phänomenen und den literarischen und physikalischen Topoi zu suchen.

Scientific GOTO Fictions – ein unbedingter Rücksprung

Zu Beginn sollen die oben angedeuteten ›Portal-Effekte‹ (von denen einer dafür verantwortlich ist, dass sich Ms. Pac-Man zeitweise nicht auf dem Bildschirm befindet) beim Namen genannt werden. Es geht um *Sprünge in den Hyperraum* (hyperspace), um *Teleportation* (›Beamen‹) und um Reisen durch *Wurmlöcher* – also um Prozesse, bei denen ein Körper von einem Ort A zum einem anderen Ort B gelangt, ohne sich jemals dazwischen aufzuhalten – und vor allem, in kürzerer Zeit als die gewöhnliche Bewegung von A nach B dauern würde.
Das Interesse◂**3** an diesen Phänomenen kommt nicht von ungefähr, sondern stammt aus der reichhaltigen Kultur- und Literaturgeschichte. Der Teleportation ähnliche Verfahren vom Verschwinden hier und Erscheinen dort gibt es bereits in der jüdischen Mythologie◂**4** oder den Erzählungen aus *Tausend und einer Nacht*.◂**5** Auch die anderen Reiseformen besitzen literarische Vor- und Nachbilder. Drei Beispiele aus drei Gattungen seien an dieser Stelle aufgrund ihrer besonderen Darstellungsweisen genannt.

Flatland

Der Mathematiker, Theologe und Lehrer Edwin A. Abbott veröffentlichte 1884 seinen Roman *Flatland*, der dem Untertitel zufolge *A romance of many dimensions* (ders. 2009)◂**6** ist, sich daneben aber bereits aufgrund der darin abgebildeten Diagramme leicht erkennbar als Traktat über die Erweiterung der euklidischen Geometrie in höherdimensionalen Räumen (eine Definition für Hyperspace◂**7**) offenbart. In Kapitel 14 beschreibt der Held aus dem zweidimensionalen Flatland, wie er den König des eindimensionalen Lineland über höherdimensionale Räume (als denjenigen, in dem er selbst lebt) aufzuklären versucht. Als dies nicht fruchtet, tritt er den Beweis an und aus Lineland hinaus:

»Mit diesen Worten begann ich meine Gestalt aus Linienland hinauszubewegen. So lange noch ein Teil von mir in seinem Herrschaftsgebiet und in seinem Blickfeld blieb, rief der König fortwährend: ›Ich sehe dich, ich sehe dich immernoch [sic]; du bewegst dich gar nicht!‹ Als ich seine Linie aber schließlich verlassen hatte, kreischte er auf. ›Sie ist weg! Sie ist tot!‹ – ›Ich bin nicht tot‹, entgegnete ich; ›Ich befinde mich bloß außerhalb von Linienland, das heißt, außerhalb der Geraden, welche Ihr als ›Raum‹ bezeichnet, und im tatsächlichen Raum, worin ich die Dinge sehen kann, wie sie sind‹« (Abbott 2009, 112).

Um der Leserin den Prozess zu illustrieren, fügt Abbot an dieser Stelle eine Grafik in seinen Text ein:

Abb. 1: Zweidimensionale Figur in/auf eindimensionaler Welt

From Travel by Wire! (to a) Wireless World

Arthur C. Clarke veröffentlichte seine erste Kurzgeschichte im Dezember 1937 in der Zeitschrift *Amateur Science Fiction Stories*. Sie heißt *Travel by Wire!* In ihr beschreibt der Erzähler, welche Probleme vor der Erfindung des »radio-transporter« beim Reisen über Drahtleitungen bestanden haben.

»You people can have no idea of the troubles and trials we had to endure before we perfected the radio-transporter, not that it's quite perfect even yet. The greatest difficulty, as it had been in television thirty years before, was improving definition, and we spent nearly five years over that little problem. As you will have seen in the Science Museum, the first object we transmitted was a wooden cube, which was assembled all right, only instead being one solid block it consisted of millions of little spheres. In fact, it looked just like a solid edition of one of the early television pictures, for instead of dealing with the object molecule by molecule or better still electron by electron, our scanners took little chunks at a time« (Clarke 2000, 1).

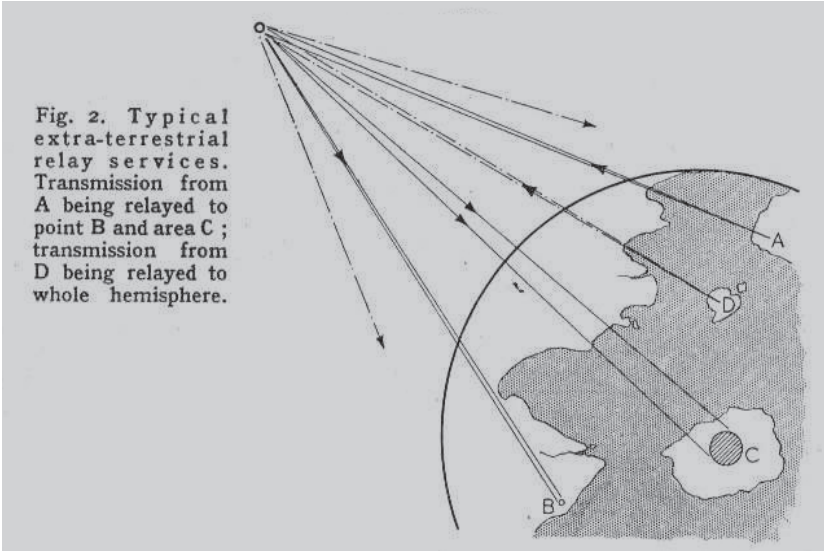


Abb. 2: Clarkes Idee bidirektionaler Satelliten-Kommunikation

Dass Clarke hier seinerzeit populäre Übertragungsmassenmedien als Sinnbilder und Namensgeber benutzt, um seine Geschichte der Erfindung drahtloser Teleportation für den Leser plastisch zu machen, kommt nicht von ungefähr. 1945 veröffentlichte er in der Zeitschrift *Wireless World* als einer der ersten Überlegungen für die Nutzung geostationärer Satelliten zur Übertragung von Informationen (vgl. Clarke 1945). Die Idee, drei auf einer (Kugelober)Fläche weit voneinander entfernte Punkte A, B und C durch einen ›Schritt nach oben‹ zu überbrücken, fügt sich nahtlos in das *Flatland*-Motiv Abbotts ein – nur dass hier anstatt der Mathematik die Medientechnologie verwendet wird, es also konkreter wird.

Die Übertragung per Satellit besitzt heute längst einen derart zeitkritischen Status, dass relativistische Effekte auf die Zeit bei Übertragungsgeschwindigkeiten nahe der Lichtgeschwindigkeit eingerechnet werden müssen. Clarkes ›crossover‹ von Science Fiction, Astronomie und Medientechnologie zeigt, wie nahe beieinander diese Diskurse liegen. Längst beschäftigt man sich auch in der Physik (vgl. Davis 2004) und auch in der Technik mit Teleportation.

Geons

Fünf Jahre nach Clarkes Kurzgeschichte erscheint in derselben Zeitschrift, in der 20 Jahre zuvor Albert Einstein und Nathan Rosen ein Gedankenspiel über Raum-Brücken vorgestellt hatten,◂**8** ein Aufsatz mit dem Titel *Geons*. In diesem greift der Physiker John Archibald Wheeler den Gedanken noch einmal auf und rechnet ihn weiter, jedoch nicht auf makroskopischer, sondern auf (sub)mikroskopischer Ebene, im Bereich der Planck-Länge ($1{,}616199 \times 10^{35}$ Meter). Dort verändert sich das Wesen der Raumzeit drastisch. Sie wird schaumartig und bildet Verzerrungen aus, die ›weit‹ entfernte Bereiche durch kurze Brücken verbinden:

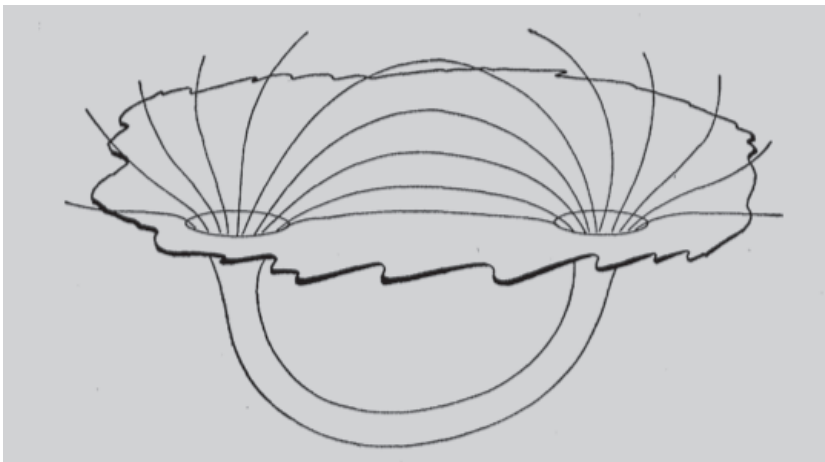


Abb. 3: Wheelers Illustration subatomarer Tunnel mit Feldlinien

»One can consider a metric which on the whole is nearly flat except in two widely separated regions, where a double-connectedness comes into evidence as symbolized in Fig. 7 [vgl. Abb. 3, S.H.]. The general divergence-free electromagnetic disturbance holding away in the space around one of these ›tunnel mouths‹ will send forth lines of force into the surrounding space, and appear to have a charge. However, an equal number of lines of force must enter the region of disturbance from the tunnel. Consequently the other mouth of the tunnel must manifest an equal and opposite charge. In such a doubly-connected space it is evidently a matter of definition whether we say that divergence-free field equations do or do not permit the existence of

electric charge. It will be convenient to say yes if the width of the tunnel is small compared to the separation of its mouths« (Wheeler 1955, 534f.).

Solche Tunnel, die Wheeler später »worm holes« (Misner/Wheeler 1957, 525) tauft, entstehen und vergehen überall und ständig im subatomaren Bereich, wenn man die Gleichungen Einsteins und Rosens von der allgemeinen Relativitätstheorie auf die Planck-Länge überträgt: »General relativity, quantized, leaves no escape from topological complexities of which Fig. 7 [Abb. 3, S. H.] is only an oversimplified symbol« (Wheeler 1955, 535). In der Nachfolge Wheelers werden diese Wurmlöcher nicht nur von der Teilchen- und Astrophysik, sondern auch von der Science Fiction aufgegriffen und im doppelten Wortsinn ausgeweitet. Wurmlöcher werden nun auch als mögliche makrokosmische Phänomene beschrieben, um sie theoretisch oder fiktional für Reisen nutzbar zu machen (Krasnikov 2011). Die Fachpublikationen zur Wurmloch-Physik sind heute beinahe nicht mehr zu zählen und die verschiedenen Theorien dazu – bislang wurde ein Wurmloch weder entdeckt, geschweige denn hergestellt◂**9** – reichen mittlerweile sogar aus, um ein Lehrbuch (vgl. Visser 1996) damit zu füllen. An den hier zitierten Quellen lässt sich neben der Tatsache, dass sie Brücken zwischen *Science* und *Fiction* zu schlagen im Stande sind, vor allem der Versuch erkennen, *Bewegungen* mit Hilfe von Abbildungen und sprachlichen Tropen *imaginierbar zu machen*. Hierzu verwenden die Autoren vorrangig Illustrationen, weil die Schrift selbst medientypisch ›statisch‹ bleiben muss, selbst wenn sie »zeitliche Abläufe räumlich auf dem Papier festzuhalten« (Coy 1993, 368) vermag. Ihr Operativwerden soll erst durch den Computerprogramm-Code zur Vollendung gelangen.

Adressen: 1961-1981

Spacewar!

Es ist eine der Gründungslegenden der Computerspielgeschichte, dass Alan Kotok, Peter Samson, Steve (›Slug‹) Russel und Dan Edwards Ende 1961 mit der Implementierung eines Digitalcomputerspiels auf dem Minicomputer PDP-1 am MIT begannen. Das Spiel Spacewar! galt dort als Verheißung einer neuen Zeit, in der Millionen Dollar teure Computerhardware endlich zur Spielplattform für Hacker werden konnte. Das Spiel folgte einfachen Regeln: Zwei gegnerische Spielerinnen steuern in Spacewar! ihre Raumschiffe um eine Sonne, die sie beständig anzieht und in die sie zu stürzen drohen. Ziel des Spiels ist es, nicht nur dieser Gravitation zu entkommen, sondern auch das Raumschiff der Gegnerin

abzuschießen. Am MIT wurde Spacewar! vor allem als programmiererische Herausforderung begeistert aufgenommen (vgl. Levy 1984, 46-57). Zu den in rascher Folge hinzugefügten Erweiterungen gehörte nicht nur die Applikation eines realistischen Sternenhimmel-Abbildes (Pias 2002, 85), sondern auch bald ein ›cheat‹, mit dem das Spiel leichter zu spielen war:

»And then there came a – startling development called Hyperspace – when your situation got desperate you could push both turn buttons at once and go into hyperspace: disappear from the screen for a few seconds and then reappear at a random new position ... maybe. ›Hyperspace was in within a month or so‹, says Russell. ›It's a little controversial. Some people deplore it, and it's fairly common to play games without it ... It was of course vital to put in problems with hyperspace. You know, when you come back into normal space from hyperspace, there is initially a small energy-well which looks amazingly like a star; if a torpedo is shot into that energy well, lo and behold [sic] the ship blows up. There is also a certain probability of blowing up as you finally break out of hyperspace. Our explanation was that these were the Mark One hyperfield generators and they hadn't done really a thorough job of testing them – they had rushed them into the fleet. And unfortunately the energies that were being dissipated in the generators at breakout were just barely what they could handle. So the probability of the generator flying apart and completely killing the spaceship was noticeable on the first couple of uses, and after four uses it was only an even chance of surviving hyperspace. So it was something that you could use but it wasn't something that you wanted to use‹« (Brand 1972).

Derartige Hyperraum-Sprünge sind nur dann für die Spielerin nachvollziehbar, wenn sie sie ›sehen‹ kann; wenn sie also in einem Moment das Raumschiff auf Position A sieht und es dort nach dem Aktivieren des Hyperraum-Sprungs verschwindet, um im nächsten Moment auf einer ungewissen Position B wieder aufzutauchen. Die kurzfristige Unsichtbarkeit der Spielfigur ist an diese prinzipielle Sichtbarkeit gebunden, weshalb die Nutzung eines Kathodenstrahl-Bildschirms (anstelle der sonst für Ein- und Ausgaben verwendeten Teletypewriter) das Spiel überhaupt erst spielbar macht, wie Pias (2002, 84f.) bemerkt.

Computerschach

Daher bildet die bildschirmgrafische Darstellung eines ganz anderen, wesentlich ›unspektakuläreren‹ frühen Computerspiels ebenfalls die Bedingung für ihre Einordnung in das genannte spatiologische Motiv: Die Rede ist vom Computerschach, das zu den Urszenen des Digitalcomputers selbst gehört, weil sich alle frühen Computerpioniere (Turing, Zuse, von Neumann, Shannon) auch mit Schachimplementierungen herumtrugen. Jahrzehntelang wurde die Rechen-

kraft der Schachcomputer und -programme lediglich zur Analyse der Stellung und zum Berechnen der Züge genutzt. Mangels Grafikdisplays und Notwendigkeit (Schachbretter und -figuren waren ja stets verfügbar) wurde das Spielgeschehen allenfalls zu programmiererischen Zwecken in so genannten »Bit Boards« (vgl. Adel'son-Vel'skii u.a. 1970, 242ff.) virtualisiert:

»a mapping of the chess board squares to the computer's internal binary structure, which allows the computer to store and analyze board positions very efficiently« (The Computer History Museum o. J.).

Diese »Bit Boards« stellen jedoch schon deshalb keine Visualisierung dar, weil man nicht (ohne weiteres) in den Speicher hinein schauen kann. Könnte man es, bekäme man zu sehen, was ab Mitte der 1970er-Jahre◂**10** bei etlichen Schachprogrammen sichtbar wurde: Dass die Spielfiguren nicht etwa ›ziehen‹, sondern ›springen‹. Anstatt die Figuren-Grafiken animiert über das Feld zu bewegen, werden Züge zu Beginn des grafikfähigen Computerschachs so dargestellt, dass die betreffende Figur auf Feld A zu blinken beginnt, um kurz darauf von dort zu verschwinden und blinkend auf Feld B aufzutauchen, wo sie sich dann wieder als Grafik manifestiert. Weil das Entstehen und Vergehen hier visualisiert wird und – anders als bei Spacewar! – Start- und Zielpunkt der Spielerin bekannt sind, scheint es nicht abwegig, hinter dieser Ortsveränderung eine Form der Teleportation zu sehen, einen gezielten Sprung – anders als beim Hyperraum-Sprung dieses Mal jedoch ins Gewisse.

Ms. Pac-Man

Die dritte Bewegungsfigur ist die vielleicht spektakulärste und am häufigsten ästhetisierte. Als populäres Beispiel wähle ich das Spiel Ms. Pac-Man (1981◂**11**), um das es im weiteren Verlauf gehen wird. In Ms. Pac-Man bekommt die Spielerin in einer Mischperspektive ein Labyrinth in Aufsicht und Spielfiguren in der Seiten- sowie Frontalansicht präsentiert. Sie steuert die Ms. Pac-Man-Figur durch dieses Labyrinth, um in den Gängen liegende Punkte einzusammeln. Dabei wird die Spielfigur von vier Geistern verfolgt, derer sie sich zwar mithilfe so genannter »Power Pills« erwehren kann (das Einsammeln einer Power Pill ermöglicht Ms. Pac-Man, die dann vor ihr fliehenden Geister zu fangen und zu zerstören), die sie bisweilen jedoch so in die Ecke drängen, dass es keinen Ausweg zu geben scheint – außer an bestimmten Stellen an den Seitenrändern des Labyrinthes, wo dieses (nach links, nach rechts) in Richtung Bildschirmrand geöffnet ist und einen Durchgang sowohl für Ms. Pac-Man als auch für die Geis-

ter gewährt. Verlässt eine der Figuren das Labyrinth durch einen linken Ausgang, taucht sie fast (!) unverzüglich an einem rechten Eingang wieder auf – und umgekehrt.
Dieser Übergang, der sich deutlich von den vorher beschriebenen Sprüngen unterscheidet, vollzieht sich in voller Bewegung und Beibehaltung der Bewegungsrichtung und es mutet an, als gäbe es einen Tunnel, der (vielleicht hinter dem Bildschirm?) von der linken zur rechten Labyrinth-Öffnung führt. Ein buchstäblicher ›Mise en Abyme‹-Effekt, der den Eindruck des geschlossenen Spiel(raum)s aufbricht und das relativitäts-physikalische Phänomen des Wurmlochs als Abkürzung zwischen zwei weit entfernten Raumpunkten ins Spiel bringt.
An dieser Stelle wechsele ich von der ›oberflächlichen‹ Betrachtung der ästhetischen Motive und Phänomene auf die ›Unterfläche‹ und wiederhole die Frage: Wo ist Ms. Pac-Man zu der Zeit, wenn sie weder an Ort A noch an Ort B zu sehen ist? Um diese Frage zu klären, widme ich mich zunächst den operativen Eigenschaften von Code-Strukturen, die ich dann am Beispiel von Ms. Pac-Man konkreter beschreibe.

Zeitreise-Diagrammatiken

Betrachtet man die Illustrationen von Abbot (Abb. 1) und Clarke (Abb. 3), so fällt auf, dass es Abbot zwar noch gelingt, auf dem ebenen Papier seines Buches die Differenz zwischen ein- und zweidimensionaler Welt grafisch darzustellen,◂**12** Clarke jedoch bereits auf Bewegungslinien mit Richtungspfeilen (Vektoren) angewiesen ist, um die Lokal- und Temporaleffekte seiner Kommunikation im Raum zu illustrieren. In der Frage nach der Verzeitlichung von Schrift und Bild als Diagramm wird Lessings Theorie des Laokoon erneut aktuell – insbesondere vor dem Hintergrund einer *operativen Diagrammatik*, wie sie die Medienwissenschaft seit einiger Zeit beschäftigt (vgl. Ernst 2006, 16f.): Lassen sich Prozesse, in denen Zeitverläufe eine Rolle spielen, auch auf der *Textoberfläche* oder nur in der Text-Semantik abbilden?
Die Computerwissenschaft, durch die die Schrift auf dem Papierstreifen in der Turingmaschine zuallererst operativ wurde, beantwortet diese Frage mit ›ja‹. Sie hat zur Illustration von Zeitverläufen ein spezifisches Diagramm ersonnen: das *Flussdiagramm* der *Programmabläufe* und *Datenflüsse*. Seine Ursprünge reichen bis in die frühen 1920er-Jahre zurück.◂**13** Es wurde in den 1960er-Jahren von der noch jungen Informatik als Abstraktionsmittel aufgegriffen:

»A flowchart is a pictorial plan showing what you want the computer to do, and *in what sequence* to do it. The flowchart improves communication between you and other people by allowing others to understand how you have instructed the computer to solve a problem« (Farina 1970, 1; Herv. S. H.)

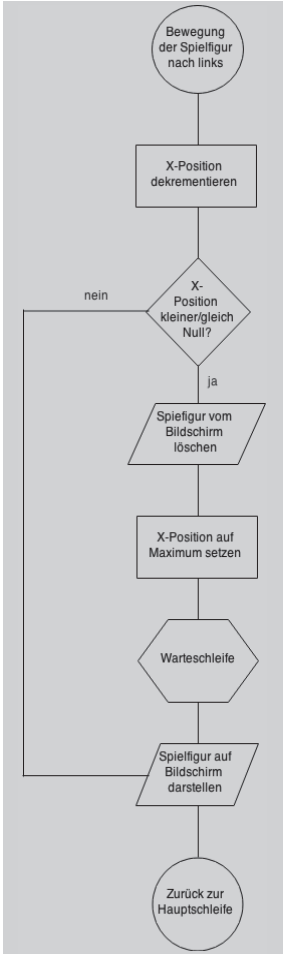


Abb. 4: Flussdiagramm ›Prüfung: Eintritt in den Tunnel‹

Solche Programmablaufpläne werden von oben nach unten oder links nach rechts gelesen und können im Unterschied zu anderen Code-Verallgemeinerungen (wie zum Beispiel dem Pseudocode◂**14**) auch Zeitprozesse visualisieren. Im Folgenden ein Beispiel für die Prüfung, ob Ms. Pac-Man das ›Wurmloch‹ betreten hat.

Die Formen der Elemente kennzeichnen deren Funktion: *Ovale* und *Kreise* stellen Kontrollpunkte, *Linien* und *Pfeile* Verbindungen, *Rechtecke* Operationen, *Rauten* Entscheidungen mit Verzweigungen, *Parallelogramme* Ein-/Ausgaben und *Rechtecke mit vertikalen doppelten Linien* Unterprogramme dar. Dadurch, dass in diese grafischen Elemente Text eingetragen wird, wird dieser zugleich zeitlich ›verflüssigt‹. Der Sinn des obigen Ablaufplans ist intuitiv erfassbar; die Linien symbolisieren den zeitlichen Ablauf des Vorgehens durch räumliche Positionierung. In Hinblick auf das Thema sind hier zunächst die Verzweigung bei der Entscheidung ›X-Position kleiner/gleich Null?‹ sowie die lange Verbindung von dort nach ›Spielfigur auf dem Bildschirm darstellen‹ interessant. An ihnen zeigt sich, dass der Programmablauf keineswegs linear verlaufen muss, sondern nach Prüfung von Bedingungen verschiedene Verzweigungen nehmen und Sprünge zu ›früheren‹ oder ›späteren‹ Programmteilen enthalten kann.

Die zeitliche Struktur lässt sich selbstverständlich auch aus dem Code selbst herauslesen, wenn die Leserin dabei die Rolle des Assemblierers einnimmt: Hier, wie in nicht-operativen Texten, ist die Anordnung ›auf den ersten Blick‹ linear. Die Linearität des Programms wird sogar durch die sukzessiv anwachsenden

Adressen erkennbar. Sie stellen jedoch gleichzeitig ›Label‹ dar, die angesprungen werden können. Sprungziele sind auf Maschinenebene immer Speicher-Adressen, an denen der nächste auszuführende Code steht.◂**15** Die menschliche Assembliererin vollzieht diese Sprünge beim Lesen von Code nach. Sprünge zu einer ›früheren‹ oder ›späteren‹ Adresse als der Nachfolgenden lassen sich bei ihr als messbare Augenbewegung erkennen (vgl. Busjahn u.a. 2011).

Jump relative

Dies lässt sich im Folgenden anhand eines Code-Teils aus dem Arcade-Automatenspiel Ms. Pac-Man nachvollziehen. Der gesamte Spielcode umfasst einen Adressraum von insgesamt 40 Kilobyte. Der Z80-Prozessor in Ms. Pac-Man ist mit 3,072 Megahertz getaktet (also circa drei Millionen Takten pro Sekunde). Der folgende 24 Byte lange Routine-Teil verbraucht 109 bis 114 Takte (je nachdem ob die Bedingungen in 0525_{HEX}◂**16** erfüllt sind oder nicht) – dauert also etwas länger als 3,5 Mikrosekunden.
Für alle Zustände (Bewegungen, Interaktionen …), die die Spiel- und Gegnerfiguren im Spiel einnehmen können, existieren Speicheradressen, aus denen sich der jeweilige Zustand auslesen und innerhalb des Programms dann als Bedingung verarbeiten lässt. Ob sich Ms. Pac-Man im Labyrinth aufhält oder gerade einen Tunnel betritt, ist in Adresse $4dbf_{HEX}$◂**17** gespeichert. Sucht man im Programmcode (Lawrence o. J.) nach dieser Adresse, findet man die Stellen, an denen dieser Zustand (Flag) geändert, also der Tunnel betreten (1) oder verlassen (0) wird. Dieser Wert wird natürlich von der X- (gespeichert in $4d3a_{HEX}$) und Y-Position ($4d39_{HEX}$) der Spielfigur determiniert.
Die folgende kleine Z80-Assembler-Routine, über deren Funktion sich der Autor◂**18** nicht ganz im Klaren ist (vgl. den überschriftartigen Einstiegskommentar), ist gut dazu geeignet, den vorangegangenen Leseprozess zu illustrieren und darüber hinaus sowohl die Programmstruktur als auch die symbolische Lokalisierbarkeit der Spielfigur zu klären.

[Adresse	OpCodes	Mnemonic◂**19**	Daten	Kommentar, S.H.]
; check for moving through a tunnel?				
051c	21 a0 4d	ld	hl,#4da0	
051f	06 21	ld	b,#21	
0521	3a 3a 4d	ld	a,(#4d3a)	
0524	90	sub	b;	pac going through a tunnel?

```
0525    20 05    jr    nz,#052c    ; (5)    ◄20
0527    36 01    ld    (hl),#01
0529    c3 8e 05 jp    #058e
052c    cd 17 10 call  #1017
[...]
058e    21 02 4e ld    hl,#4e02
0591    34       inc   (hl)
0592    c9       ret                        ◄21
```

Zunächst soll das Fragment Adresse für Adresse in seiner Funktion kommentiert werden:

$051c_{HEX}$: Die Routine beginnt damit, dass das Register hl mit dem Hexadezimalwert $4da0_{HEX}$ geladen wird. Dieser Wert stellt eine Adresse im Scratchpad-RAM (siehe unten) dar, die im weiteren Verlauf arithmetischen Operationen unterzogen wird, um eine neue Adresse (ein Sprungziel) zu berechnen.◄**22**

$051f_{HEX}$: Das Register b wird mit dem Wert 21_{HEX} geladen.

0521_{HEX}: Der Inhalt der RAM-Adresse $4d3a_{HEX}$ wird in das Register a◄**23** geladen. (In $4d3a_{HEX}$ ist die X-Position von Ms. Pac-Man gespeichert.)

0524_{HEX}: Der Inhalt von Register b (siehe $051f_{HEX}$) wird vom Inhalt des Registers a subtrahiert. Durch diese Subtraktion wird also die X-Position von Ms. Pac-Man verändert. Sie rückt nach links – um 21_{HEX}-Bildschirm-Adressen.

0525_{HEX}: Wenn diese Subtraktion nicht das Ergebnis Null (nz: non zero) hatte (Ms. Pac-Man also noch nicht am linken Bildschirm-Rand angekommen ist), wird 5 Adressen nach vorn (zu $052c_{HEX}$) gesprungen.

0527_{HEX}: In die RAM-Adresse, die im Register hl gespeichert ist (siehe $051c_{HEX}$), wird der Wert 1 abgelegt. Hierbei scheint es sich um ein Flag zu handeln, das anzeigt, ob Ms. Pac-Man den Tunnel betritt.

0529_{HEX}: Es wird direkt ans Ende der Routine (nach $058e_{HEX}$) gesprungen.

$052c_{HEX}$: Eine Unterroutine, die ab Adresse 1017_{HEX} startet, wird aufgerufen. Ihr Zweck ist (noch) nicht eindeutig klar. Nach Ablauf dieser Routine wird in die folgende Adresse ($052f_{HEX}$) mit ret zurückgekehrt.

$052f_{HEX}$-$058d_{HEX}$: Hier werden verschiedene Subroutinen mit call aufgerufen und der Wert für die X-Position von Ms. Pac-Man weiter bearbeitet, um eine kontinuierliche Bewegung zu generieren.

$058e_{HEX}$: Das Register hl wird mit dem Hexadezimalwert $4e02_{HEX}$ geladen. Dieser repräsentiert die Adresse, in der der Levelfortschritt gespeichert ist.

0591_{HEX}: Der Inhalt der Adresse $4e02_{HEX}$ (siehe $058e_{HEX}$) wird um 1 erhöht. Ms. Pac-Man ist dem Levelende also einen Schritt näher gekommen.

0592_{HEX}: Dieses Unterprogramm ist beendet. Es wird an die Adresse zurückgekehrt, von wo es aufgerufen wurde.
Ich möchte das Augenmerk zunächst auf die bedingten Sprünge in 0525_{HEX}, 0529_{HEX} und $052c_{HEX}$ lenken. Auf der Maschinenebene bedeuten solche Sprünge, dass dem internen Programmzähler eine Adresse übergeben wird, die vom ›normalen Ablauf‹ des Programms (nämlich einfach der Adresse des nachfolgenden Befehls) abweicht.◂**24** Dies führt dazu, dass das Programm an dieser neuen Adresse weiter ausgeführt wird. Maschinenprogramme werden taktweise abgearbeitet. Wird durch einen Sprungbefehl eine Adresse weiter vorn oder hinten im Programm auf den Programmzähler gelegt, dann findet bereits *beim nächsten Takt* der Sprung dorthin statt. Das bedeutet, dass innerhalb von etwa 325 Nanosekunden eine jede beliebige Adresse im Speicher angesprungen werden kann.
Dieser Prozess ist absolut nichts Ungewöhnliches◂**25** und wird innerhalb von Ms. Pac-Man hundertfach ausgeführt. Dennoch scheint mir die Bemerkung angebracht, dass Sprünge, die in den ›Wurmloch‹-Routinen des Codes ausgeführt werden, auch metaphorischen Charakter besitzen – oder anders herum: Die Existenz des Wurmloch-Motivs auf der Oberfläche von Ms. Pac-Man erscheint vor diesem Hintergrund als eine emblematische Verdopplung jenes Prozesses, der zugleich auf der Unterfläche des Spiels stattfindet. Nachdem der Sprung nun in seinen diagrammatischen, symbolischen und strukturalistischen Facetten vollzogen wurde, möchte ich zum Schluss ins Reale zurückkehren und mich der Hardware von Ms. Pac-Man zuwenden.

Jump absolute

Die Grafiken, die sich auf dem Bildschirm des Ms. Pac-Man-Spielautomaten zeigen, stellen eine Komposition verschiedener Elemente dar. Die *Formen* all dieser Elemente sind im Character-ROM gespeichert. Die Bildschirm-Hintergrundgrafik (das Labyrinth) besteht aus ›Tiles‹, sprich unterschiedlichen Grafik-Mosaikelementen, die so angeordnet werden, dass sie ein zusammenhängendes Bild ergeben. Vor bzw. zwischen diesen bewegen sich die Spielfiguren (Ms. Pac-Man, die Geister) als ›Sprites‹, sprich Grafikelemente, die unabhängig vom Hintergrund animiert werden können, zu denen sich jedoch Informationen darüber abfragen lassen, ob sie miteinander oder mit einem Hintergrund-Element (Früchte, Punkte, Power Pills) kollidiert sind. Schließlich werden noch Schriftzeichen angezeigt, die als Zeichensatz (Characters) fest definiert vorliegen und wie ›Tiles‹ auf dem Bildschirm positioniert werden.

Die Darstellung des zellulären Aufbaus des nebenstehenden Sprites (Abb. 5) lässt sich zugleich als Diagramm für seinen ›Speicherplatz‹ lesen, denn jedes Pixel belegt ein Bit – in unterschiedlichen Speichern der Hardware. Die Menge der Spielfiguren-Bits gibt ihre ›Größe‹ (sowohl ihre grafische Dimensionierung als auch ihren Speicherverbrauch) an. Dieser Platzbedarf und seine Befriedigung sollen zunächst diskutiert werden. Ms. Pac-Man existiert, wie schon der Vorgänger Pac-Man in einer Vielzahl von Implementationen für Spielkonsolen, Heimcomputer, Handhelds und als Arcade-Automat. Der Hardware von letzterem, aus dem auch das obige Codefragment disassembliert wurde, möchte ich mich nun zuwenden. Neben dem Gehäuse, dem Monitor (und seiner Steuerelektronik), dem Eingabepanel (Münzeinwurf, Joystick, Funktionstasten) ist es vor allem das *Mainbaord*, das ein

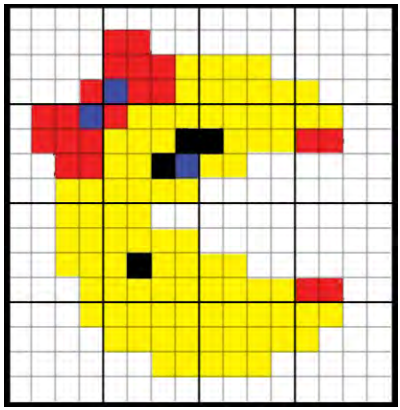
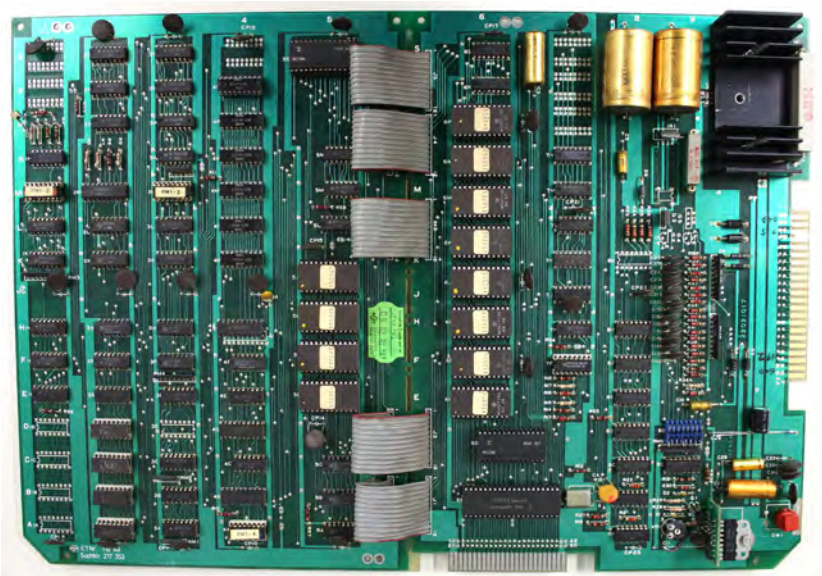


Abb. 5: Das Ms. Pac-Man-Sprite besteht aus 16 mal 16 Pixeln (256 Bit, also 32 Byte)

Abb. 6: Das »(Ms.) Pac-Man«-Board



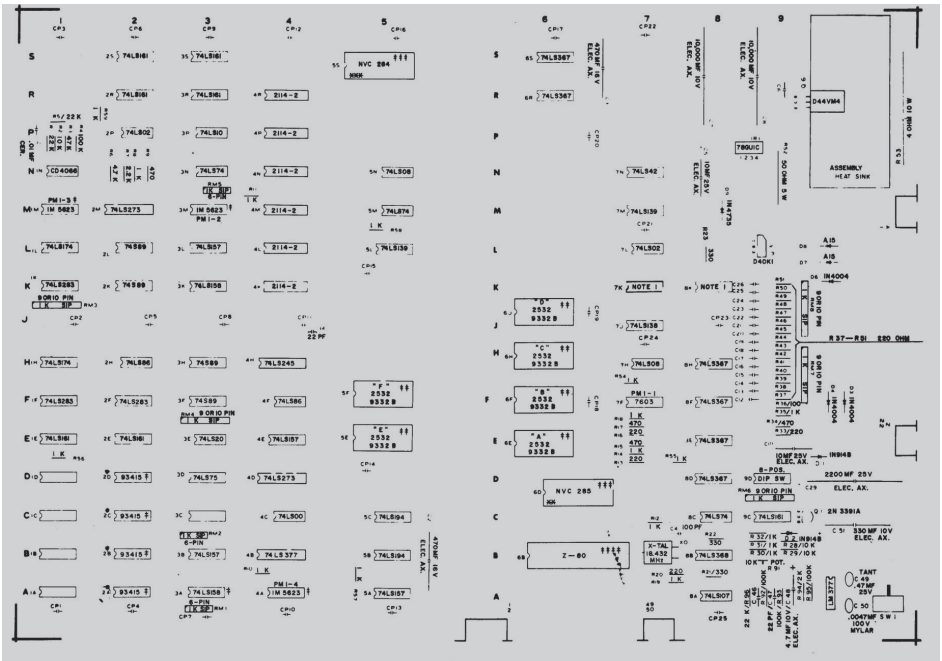


Abb. 7: Das Board zwischen Bestückungsplan und Diagramm

Arcade-Computerspiel definiert. Auf ihm befinden sich – im Fall von Ms. Pac-Man – der Mikroprozessor, die ROMs mit der Software und vordefinierten Daten, RAM-Bausteine als Grafik- und Zwischenspeicher sowie verschiedene Steuer- und Logikbausteine, die für die Kooperation der Hardware-Bestandteile zuständig sind (vgl. Abb. 6). Für Service- und Reparaturzwecke◂**26** ist dieses Board rasterartig aufgebaut und symbolisch in Zeilen (A-S) und Spalten (1-9) aufgeteilt.

Drei Elemente darauf verdienen unser Augenmerk, wenn es um die Frage des realen Ortes von Ms. Pac-Man geht: Die *EPROMs (erasable programmable ROM) mit dem Programmcode*. Sie befinden sich in 6E-J (bei kleiner dimensionierten ROMs zusätzlich auch bis 6K-N) und besitzen jeweils 4096 Byte Kapazität. Die *EPROMs mit den Characters/Zeichen/Spielfiguren* befinden sich in 5E und 5F.◂**27** Das *allgemeine RAM* als Speicher für Berechnungen, Zwischenspeicher usw. in 4K-M. Diese letzten sechs jeweils 512 Byte großen Bausteine unterscheiden sich noch einmal in ihrer Funktion für das Spiel: 4M und 4R bilden das *Scratchpad-RAM*. 4K und 4N bilden das *Video-RAM*. 4L und 4P bilden das *Color-RAM*. Diese

RAM-Bausteine werden verschiedenen Adressen zugeordnet. Das Scratchpad-RAM liegt in den Adressen $4C00_{HEX}$-$4FF1_{HEX}$, das Color-RAM bei 4400_{HEX}-$47FF_{HEX}$ und das Video-RAM 4000_{HEX}-$43FF_{HEX}$.

Damit zeigt sich bereits, dass die Ms. Pac-Man-Spielfigur wie alle andere Computergrafik auch keineswegs nur Software ist, sondern stets abhängig von dezidierter Hardware, die durch ganz unterschiedliche Elemente an ganz verschiedenen ›Orten‹ zu ihrem Gestalt-Effekt beiträgt: durch ihre *symbolische Position* in den EPROMs (für den Programmcode), im Scratchpad-RAM (als Informationsspeicher des Programms) sowie in den Registern des Mikroprozessors (welcher selbst zugleich der Strukturspeicher des Spiels ist), durch ihre *grafische Form* (im Character-ROM) übertragen auf eine *Positionsadresse* auf dem Bildschirm (im Video-RAM, nicht zu verwechseln mit dem speziell für die Bild*ausgabe* konzipierten VRAM!) und überlagert mit ihren *Farben* (im Color-RAM). Sie alle sind notwendig, damit ›Software‹ sichtbar wird. Der Bildschirm selbst produziert diese Gestalt dann allerdings auch nicht als geschlossene Form, sondern löst sie in Rasterzeilen auf, die ihre Entsprechung in den Adressen des Video-RAMs haben.

Suchen wir uns aus ›phänomenologisch naheliegenden Gründen‹ das Video-RAM als Repräsentant des Ortes der Ms. Pac-Man-Figur heraus, können wir den Sprung auf der Adressbasis nachvollziehen, den sie leistet, wenn sie sich in das linke untere ›Wurmloch‹ begibt und aus dem rechten unteren wieder erscheint: Video-RAM-Adresse $43B8_{HEX}$ ist die direkt am Bildrand gelegene Adresse innerhalb des unteren linken Wurmloch-Durchgangs; 4058_{HEX} die entsprechende gegenüber am rechten Bildrand. Die Distanz zwischen beiden beträgt also 360_{HEX} Adressen. Die nachfolgende Abbildung 8 zeigt den Zusammenhang zwischen der dargestellten Grafik und ihrer Position im Video-RAM:

Diese Distanz wird durchaus nicht nur symbolisch (rechnerisch bzw. algorithmisch) übersprungen, sondern auch im Realen. Die beiden Video-RAM-Bausteine fassen wie geschrieben jeweils 512 Byte. Der gesamte Videospeicher ist genau doppelt so groß, weshalb zwei dieser Bausteine (4K und 4N) benötigt werden. Das Video-RAM an 4K enthält die Adressen 4000_{HEX}-4200_{HEX}, das Video-RAM an 4N die Adressen 4201_{HEX}-$43FF_{HEX}$. Der Sprung findet also nicht bloß innerhalb eines integrierten Schaltkreises, sondern ganz profan zwischen den Bausteinen 4K und 4N statt.◂**28** Der Eingang der Spielfigur in den unteren linken und ihr Erscheinen am rechten unteren Wurmloch-Durchgang entspricht also auf der materiellen Ebene der Spielhardware tatsächlich einer Distanzüberwindung: vom RAM-Baustein 4K (ihrem Ursprung) in den Mikroprozessor (der ›exotischen Zone‹ ihrer Verrechnung) und von dort zum RAM-Baustein 4N (ihrem Ziel).

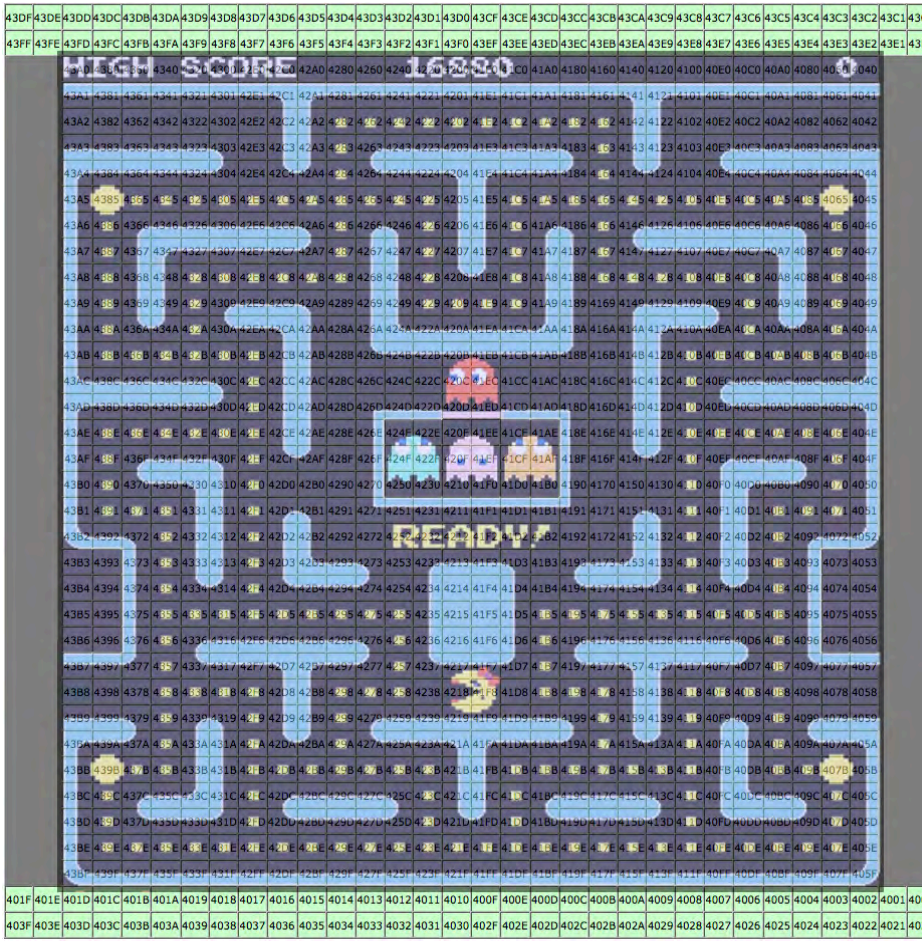


Abb. 8: Das Video-RAM, überlagert mit einem Level des Spiels
(jede Gitterzelle entspricht 32 Byte Video-RAM)

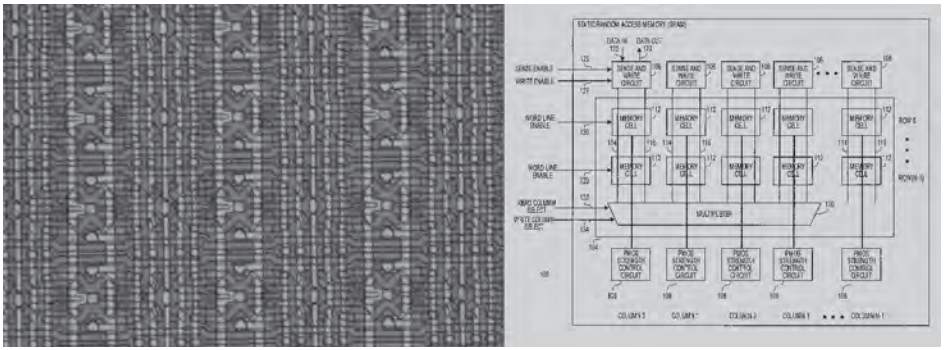


Abb. 9 : Links: Mikroskopische Aufnahme einer SRAM-Chipoberfläche¯30; Rechts: Prinzipschalt-Diagramm eines SRAM-Speichers

Quantensprünge

Der letzte Schritt zum Ort der Spielfigur führt uns auf die Silizium(ober)fläche des RAM-Speichers selbst. Auf ihm sind die Flip-Flops gitterartig angeordnet – dieser Aufbau◂**29** ist ein direktes Erbe des Magnetkernspeichers. Was sich im ›Großen‹ der Adressierung zwischen Mikroprozessor und ROM-/RAM-Speichern abspielt, wiederholt sich in der Mikroelektronik der Speicher noch einmal.

Bei den beiden Video-RAMs handelt es sich um Static-RAM-Bausteine (SRAM) des Typs »2114-2« (vgl. Intersil o. J. und Völz 2007, 184-187). Diese speichern ihre Informationen bitweise durch Flip-Flops. Eine Adresse bezieht sich dabei stets auf zwei 4-Bit-Flip-Flop-Gruppen, in denen also zusammen 1 Byte Daten gespeichert sind. Beim Zugriff des Mikroprozessors auf solch eine Adresse findet zunächst ein Chip-Select statt, der aufgrund der Adresse den benötigten RAM-Baustein auswählt. Über den Adresskodierer wird dann ein Row- und Column-Select ausgeführt, mit dem die zwei zur Adresse gehörigen 4-Bit-Einheiten (Low Nibble, High Nibble) auf dem SRAM lokalisiert und ausgewählt werden. Diese werden sodann seriell ausgelesen, durch einen Seriell-Parallel-Wandler (Multiplexer) zu einem 8-Bit-Datensatz zusammengefügt und an den Datenbus des Prozessors übertragen, der damit weiter operiert. Um die gesamte Ms. Pac-Man-Spielfigur vom Prozessor ins Video-RAM zu schreiben, muss dieser Vorgang 32mal wiederholt werden.

Der »2114-2«-RAM-Baustein ist in NMOS-Bauweise konstruiert (bestehend aus n-Kanal-Metall-Oxid-Halbleiter-Feldeffekttransistoren, vgl. Rost 1966, 140-151 und Malone 1995, 78-80). Seine Transistoren werden durch einen aufwendigen fotolitografischen und chemo-physikalischen Prozess direkt in eine Silizium-Oberfläche (Wafer) implementiert (Integrierter Schaltkreis). Ein Transistor ist ein schaltbarer elektrischer Halbleiter. Der Schaltprozess geschieht bei ihm voll elektronisch, indem zwei voneinander entfernte (isolierte) Kontakte Source und Drain in einem dotierten Halbleiter eingebettet sind, der selbst einen dritten Anschluss (Gate) besitzt. Beim Anlegen einer Spannung an das Gate verändert sich die Leitfähigkeit des Halbleiters vom Nichtleiter hin zu einem Leiter durch Verkleinerung der zuvor isolierend wirkenden Ladungszone (vgl. Malone 1995, 64). Die elektrische Leitfähigkeit in Feststoffen/Kristallen entsteht dadurch, dass Elektronen zwischen zwei unterschiedlich hohen Energieniveaus (Bändern) hin und her springen können. Auf dem Oszilloskop zeigt sich, dass dieser Schaltprozess nicht instantan geschieht, sondern eine (wenn auch sehr steile) Flanke zwischen Null Volt und (hier) +5 Volt (beim »2114-2« sind das 200 Nanosekunden) besteht. Auf der Quantenebene fließen die Elektronen jedoch nicht, sondern befinden sich auf distinkten Energieniveaus in den Orbitalen des Siliziumatoms und wechseln innerhalb des Kristallgitters vom Valenz- in das Leitungsband durch so genannte »Übergänge« (Nils Bohr hatte dafür 1913 den Begriff des »Quantensprungs« geprägt, vgl. Wolf 1990, 94-102). Bei diesem ebenfalls nicht instantan vollzogenen Vorgang ›überspringen‹ sie Bandlücken, welche »quantenmechanisch verbotene Zonen« darstellen (vgl. Rost 1966, 108ff.).

An dieser Stelle können wir wieder an die ›Oberfläche‹ des Spiels und seiner Grafik zurückkehren, denn derselbe Effekt ist auch für das Erscheinen eines Leuchtpunktes auf dem Bildschirm verantwortlich: Dort, wo ein Elektron aus dem Kathodenstrahl auf die Monitor-Innenseite trifft, regt es ein Elektron des dort aufgebrachten Leuchtstoff-Moleküls an. Dieses ›springt‹ auf ein höheres Energieniveau und gibt beim Rücksprung auf sein ursprüngliches Niveau langwelliges (sichtbares) Licht ab. Je nach chemischer Beschaffenheit des Beschichtungsmaterials leuchtet ein unterschiedlich farbiger Lichtpunkt auf. Wollte man der Homologie folgen, könnte man den Topos des Sprungs durch exotische Zonen in Ms. Pac-Man also bis auf die Quantenebene verfolgen.

Sprungziel

Die zuletzt geschilderten Prozesse finden in nicht mehr sinnlich wahrnehmbaren Zeitabständen und technisch nicht mehr messbaren Größenordnungen◂**31** statt. Sie sorgen dafür, dass der Computer als zeitbasiertes Medium mit seiner getakteten Eigenzeit durch seine Rechengeschwindigkeit in der Lage ist, nicht nur makroweltlich existierende Phänomene in Hard- und Software nachzubilden, sondern auch Phänomene und Vorgänge, die gar nicht wirklich existieren. Hierin zeigt sich einer der wesentlichen Gründe für die Verwendung von Computern zu Simulationszwecken – beispielsweise exotischer physikalischer Prozesse, die mit anderen Medien nicht darstellbar wären (wie >3-dimensionale Objekte◂**32**) oder Hyperspace-, Teleportations- und Wurmloch-Effekte.
Dass in Computerspielen diese physikalischen Exoten zu Motiven und sogar zu Elementen des Gameplay werden, stellt eine Eskalation dieses Vermögens dar, denn wenn jedes Computerprogramm auch immer ›bloß ein Spiel‹ ist (Computer können gar nichts anderes als Situationen symbolisch durchzuspielen), dann sind Computerspiele die ›ehrlichste‹ Art von Computerprogrammen, die das ludische ›Was wäre wenn?‹ der Simulation durch implementierte Interaktivität auf die Spitze treiben. Im vorliegenden Fall bringt das Spiel auf der Basis von operativer Schrift und Mikroelektronik durchspielbare Diagramme auf den Bildschirm, bei denen ein Blick unter die phänomenale Erscheinung ihrer Oberflächen Analogien und Homologien zu seinen Ästhetiken anbietet.
Die Darstellung solcher Effekte ist natürlich nicht allein Computern vorbehalten. Wurmlöcher, Hyperspace und Teleportation sind ein populäres Motiv in der Science-Fiction-Literatur und dem Science-Fiction-Film. Der maßgebliche Unterschied zwischen diesen Motiven und der Adaption in Computern und Computerspielen zeigt jedoch zugleich, wie radikal anders dieses Medium seinen Vorläufern gegenüber ist: Wo Literatur, Film, Comic und andere die Trennung zwischen erzählter Zeit und Erzählzeit stets wahren müssen und den Bruch der erzählten Zeit (durch Montage, Vorgriff, Retrospektion ...) stets nur in ihrer Diegese vollziehen und ihre materielle Oberfläche davon nicht affiziert wird, obliegt dem Computer die Manipulation beider Zeitachsen. Er zeigt nicht nur den Sprung – er springt selbst auf Basis der in ihn einprogrammierten ›Narration‹ (namens Programmcode) und der in ihm implementierten ›Sprache‹ (namens elektronische Schaltung). Damit liefert der Computer beim Abarbeiten des Spielcodes zugleich einen Einblick in sein medientechnisches Apriori.
Was Spiele wie Portal inszenieren, ist also neben ihrem Motiv auch eine Epistemologie ihrer medialen Verfassung. Dies könnte sich natürlich auch an Portal selbst zeigen lassen, denn wie jedes Programm auf einer von Neumann-

Computerarchitektur nutzt es die oben beschriebenen Sprünge, um diese auf seiner Oberfläche zu inszenieren. Insofern stellt der Rückgriff auf Ms. Pac-Man eine (didaktische) Reduktion dar, die sich problemlos auf neue Plattformen und Codes erweitern ließe, dann jedoch ein wesentlich größeres Maß an Komplexität vorfinden würde. Im Sinne einer ›grundlagenforschenden‹ Medienarchäologie sollte Ms. Pac-Man als Beispiel für eine solche Analyse genügen. Das Wurmloch stellt zugleich ein passables Sinnbild der hier geleisteten medienarchäologischen Archivarbeit dar:

»The potential transmissibility of almost all events means that the route to primetime is more akin to what physicists call a ›wormhole‹ – a shortcut connecting distant points in space and time – than to the conventional sociological understanding of a linear bureaucratic process« (Frosch/Pinchevski 2009, 303).◀**33**

Diese Verkürzung von Distanzen und Zeiträumen ist auch das Ansinnen einer ›Archäologie der Gegenwart‹, wie sie die Medienarchäologie darstellt. In der *Verkürzung* von Textfunden auf ihre diagrammatischen (Un)Möglichkeiten, der *gerafften* Motivgeschichte von Computerspielen auf die Frage nach ihren Ästhetiken des Transports und im *Brückenschlag* zwischen Code- und Hardware-Ebene eines Computerspiels ergeben sich Erkenntnisse, die über die ästhetische, soziologische, historische, aber auch die informatische und elektrotechnische Ebene hinaus gehen.

Anmerkungen

01▶ Kittler (2002) handelt vom »Was-Sein von Computerbildern« (ebd., 178). Ich will mich im Folgenden auf deren ›Wo-Sein‹ konzentrieren.

02▶ Denn das Computerspiel ist ein zeitbasierter Medieninhalt; der Aufenthaltsort ist daher immer auch abhängig von der Zeit des Spiels. Computergrafik ist im ästhetischen wie technischen Sinne chronotopisch.

03▶ Diese Arten der Fortbewegung haben das allgemeine Interesse zumindest so weit geweckt, dass in der Wikipedia-Enzyklopädie verschiedene Einträge zu deren fiktionalen Adaptionen existieren. So gibt es eine Liste zu »Video game characters who can teleport« [http://en.wikipedia.org/wiki/Category:Video_game_characters_who_can_teleport], letzter Abruf: 30.04.2014; »Wormholes in Fiction« [http://en.wikipedia.org/wiki/Wormholes_in_fiction], letzter Abruf: 30.04.2014; »Portals in fiction« [http://en.wikipedia.org/wiki/Portals_in_fiction], letzter Abruf: 30.04.2014; und »Hyperspace ([in] science fic-

tion)« [http://en.wikipedia.org/wiki/Hyperspace_%28science_fiction%29], letzter Abruf: 30.04.2014. Eine besondere Verführung der genealogischen Art wäre es, den Spuren, die Physik und Mathematik in solchen Fiktionen hinterlassen haben, nachzugehen.

04► Im Talmud beschreibt *kefitzat ha-derekh* die Fähigkeit des Reisenden, Wege räumlich zu verkürzen und auf diese Weise schneller zu reisen. [http://www.pantheon.org/articles/k/kefitzat_ha-derekh.html], letzter Abruf: 01.09.2014.

05► In *Tausendundeine Nacht* gehört es zu den Fähigkeiten der Dschinns, ohne Zeitverlust beliebige Strecken zu überbrücken. In der Geschichte *Aladdin und die Wunderlampe* findet eine solche Reise zwischen China und Marokko statt. Vgl. [http://www.gutenberg.org/files/14221/14221-h/14221-h.htm], letzter Abruf: 01.09.2014.

06► Die englische Originalfassung findet sich online unter: [http://www.geom.uiuc.edu/~banchoff/Flatland/], letzter Abruf: 30.04.2014.

07► Zu mathematischen Hyperräumen in ihrer Geschichte und ihren Themenfeldern s. Wicks (1991).

08► Diese so genannten Einstein-Rosen-Brücken ergeben sich aus der Mathematik der Allgemeinen Relativitätstheorie und sind daher rein theoretische Gebilde. Ihre mathematische Herleitung beschließen die namensgebenden Autoren daher schon beinahe entschuldigend: »In any case here is a possibility for a general relativistic theory of matter which is logically completely satisfying and which contains no new hypothetical elements« (Einstein/Rosen 1935, 77).

09► »Die Spannung, durch die das Wurmloch am Einsturz gehindert werden soll, muss mindestens 10^{17}-mal größer sein als die Dichte der Substanz, mit der das Wurmloch gebaut wird. Ein solches Material ist bislang unbekannt« (Vaas 2013, 176).

10► Als konkretes Beispiel wäre hier Micro Chess (1976, Peter R. Jennings) zu nennen, das für Computer mit Monitor (Apple II, TRS-80 Model 1) eine wie oben beschriebene Spielfeldgrafik präsentierte. Das Phänomen eskaliert in der Arcade-Schach-Spielreihe Achron (1983f.), in der die Teleportation von Figuren als regulärer Zug ausgewiesen ist.

11► Von Midway publiziert, war es ursprünglich ein Hack des Original Pac-Man-Spiels, weshalb es auf derselben Platine basiert und große Ähnlichkeiten in der Software aufweist: »Ms. Pac-Man shares a great deal of code with the original Pac-Man« (Hodges 2008).

12► Im weiteren Verlauf, wenn der Flatlander von einem Kugelwesen in die dritte Dimension eingeladen wird, steht die Grafik des Buches jedoch vor den üblichen Problemen perspektivischer Darstellungen (vgl. Abbott 2009, 127; 145).

13► Seine Geschichte wäre in jedem Fall aber noch zu schreiben (vgl. Gilbreth/Gilbreth 1921).

14► Pseudocode ist keine Programmiersprache, sondern versucht mit natürlichsprachlichen Mitteln und formalsprachlicher Strukturierung Algorithmen gleichzeitig für den Menschen lesbar und für eine spätere Implementierung in eine Programmiersprache adaptierbar zu machen.

15► Den Aufbruch dieser scheinbaren Linearität des Codes durch Sprünge visualisiert Ben Fry

innerhalb seines »distellamap«-Projektes für die Atari-VCS/2600-Adaption von Portal, indem er Linien von jeder Sprungquelle zum Sprungziel in den Code einzeichnet: [http://benfry.com/distellamap/150dpi/pacman-illus-150dpi.png], letzter Abruf: 10.05.2014.

16 ▶ In Assembler-Sprachen werden Zahlen häufig im hexadezimalen System angegeben. 1632_{HEX} entspricht der Dezimalzahl 5682_{10}. Im Folgenden kennzeichne ich das Zahlensystem mit einer tiefergestellten 2 (binär), 10 (dezimal) und 16 (hexadezimal). Bei der Code-Analyse beziehe ich mich auf Angaben und Werte in Zaks (1987).

17 ▶ »1=pacman about to enter a tunnel, otherwise 0« (siehe Kommentar-Vorspann in Lawrence o. J.).

18 ▶ Lawrence hat das Spiel nicht programmiert, sondern disassembliert und kommentiert. Mit anderen Worten: Er hat das Reale des Computerspiels in symbolisch lesbare Form – einen Text – gebracht, was allgemein als Ausweis für Autorschaft gelten kann.

19 ▶ Wolfgang Coy erinnert daran, dass die Assembler-Mnemonics ihren Ursprung in der griechischen Gedächtniskunst hatten: »Eine Anordnung, die der Zeit, wird durch eine andere Ordnung, die des Raums simuliert« (Coy 1993, 367).

20 ▶ Im Sourcecode steht an dieser Stelle der Befehl »052c« (eine konkrete Adresse); »jr« (jump relative, Opcode 20_{HEX}) verlangt aber eine Distanz (von -127 bis +128, also: Springe von hier soundsoviele Adressen vorwärts oder rückwärts). Die ›Distanz‹ $052c_{HEX}$ wäre hier illegal groß. Hier hat der Disassembler den relativen Sprung um 5_{HEX} Adressen nach vorn (in die Adresse $052c_{HEX}$) falsch als absolute Adresse übersetzt. Auf der Ebene der Opcodes, die ja letztlich entscheidend für die Ausführung sind, ist die Zeile aber korrekt. Offenbar soll der Kommentar »(5)« genau dies angeben. Diesen Umstand, dass relative Adressierung mit dem so genannten Zweierkomplement arbeitet, zu erwähnen lohnt angesichts des Themas auch deshalb, weil sich hier der Zahlenstrahl als gebogen darstellt: Ist ein Byte voll gezählt (bei FF_{HEX} angekommen), enthält es nach der Addition von 1 den Wert 0. Hier findet also ein ›Rücksprung‹ vom einen zum anderen Ende des Zahlenstrahls statt.

21 ▶ Lawrence o. J.

22 ▶ Lawrence (o. J.) gibt im Einführungskommentar an, dass $4da0_{HEX}$ einen Status-Wert des roten Geistes speichert. Das ergibt zumindest an dieser Stelle keinen Sinn. Entweder liegt ein Interpretationsfehler vor oder $4da0_{HEX}$ wird von mehreren Programmteilen unterschiedlich genutzt (was den Code unnötig obfuskieren würde).

23 ▶ Das Register a, der ›Akkumulator‹, ist beim Z80-Mikroprozessor und vielen anderen 8-Bit-Prozessoren das zentrale Register, in dem arithmetische und logische Operationen durchgeführt werden. Da sich diese Operationen zumeist implizit und gezwungenermaßen in Register a abspielen, ist es wichtig, vor solchen Operationen erhaltenswerte Inhalte von a in einem anderen Register zwischenzuspeichern.

24 ▶ Bei Sprüngen in Subroutinen wie in $052c_{HEX}$ wird zusätzlich auch die Rückkehradresse in einem Stapelspeicher hinterlegt.

25 ▶ Es ist sogar ein konstitutiver Bestandteil von Turingmaschinen, bedingte und unbedingte

Sprünge im Code ausführen zu können.

26► Zur Lokalisierung bestimmter Elemente des Spiels lässt sich die Fehlfunktion einzelner Speicherbausteine nutzen. Offizielle wie inoffizielle ›Troubleshooting‹-Anleitungen verfahren oft so, dass sie einen sichtbaren Fehler in der Darstellung des Spiels einer möglichen Fehlerquelle (einem Baustein) zuordnen (vgl. King o. J.).

27► Hier ist die einzige *reale* Stelle, an der sich Pac-Man von Ms. Pac-Man unterscheidet. Alle anderen Unterschiede sind ›bloß symbolisch‹ (also im unterschiedlichen Programmcode) (vgl. N. N. 2000).

28► Beide Bausteine sind auf dem Board zwar nur 1 cm voneinander entfernt aufgelötet, aber nicht direkt miteinander verbunden, sondern über einen Adress-Baustein, der wiederum mit dem Mikroprozessor verbunden ist. Die ›Wege‹ zwischen den beiden RAM-Bausteinen werden von den Signalen auf den Leiterbahnen der Platine zurückgelegt.

29► Ich möchte die Tatsache, dass sich die Matrix der Spiellabyrinth-Grafik von Ms. Pac-Man über den schachbrettartigen Aufbau der Platine (mit Spalten und Zeilen) und die rasterförmige Organisation der Bit-Speicher im SRAM-Chip bis hin zur Gitterstruktur des Silizium-Kristalls selbst durchschlägt, nicht unerwähnt lassen, sie aber aus methodischen Gründen auch nicht überbetonen.

30► Für die Aufbereitung des 2114-2-RAM-ICs sowie dessen laserfotografische Abbildung danke ich der *Arbeitsgruppe Neue Materialien* von Frau Prof. Dr. Saskia F. Fischer (Institut für Physik der Humboldt-Universität zu Berlin), namentlich Herrn Jürgen Sölle.

31► Die Beobachtung von Vorgängen auf der Quantenebene beeinflusst die Vorgänge (Heisenberg'sche Unschärfe), vgl. Wolf 1990, s. 11.

32► Ein eindrucksvolles Beispiel hierfür ist die Animation »Hypercube«, bei der ein 0- bis 6-dimensionales Objekt animiert wird: [https://www.youtube.com/watch?v=-x4P65EKjto], letzter Abruf: 30.04.2014.

33► Für diesen Hinweis danke ich Wolfgang Ernst.

Literatur

Abbott, Edwin A. (2009): Flächenland. Ein Märchen mit vielerlei Dimensionen. Karlsruher Virtueller Katalog: RaBaKa Publishing.

Adel'son-Vel'skii, G. M. / Arlazarov, V. L. / Bitman A. R. / Zhivotovskii A. A./ Uskov A. V. (1970): Programming a computer to play chess. In: Russian Mathematical Surveys, Volume 25, Number 2, S. 221-262. [http://iopscience.iop.org/0036-0279/25/2/R07], letzter Abruf: 30.04.2014.

Brand, Stuard: Spacewar. Fanatic Life and Symbolic Death among the Computer Bums. In: Rolling Stone, No. 123, 7. Dezember 1972. [http://wheels.org/spacewar/stone/rolling_

stone.html], letzter Abruf: 30.04.2014.

Busjahn, Teresa/Schulte, Carsten/Busjahn, Andreas (2011): Analysis of Code Reading to gain more Insight in Program Comprehension. In: Proceeding – Koli Calling '11. Proceedings of the 11th Koli Calling International Conference on Computing Education Research, S. 1-9.

Clarke, Arthur C. (1945): Extra-terrestrial Relays – Can Rocket Stations give World-wide Radio Coverage? In: Wireless World. Radio and Electronics, 1/6, Vol. 11, No. 11 (November 1945), S. 305-308. [http://lakdiva.org/clarke/1945ww/], letzter Abruf: 30.04.2014.

Clarke, Arthur C. (2000): Travel by Wire! In: Ders.: The Collected Stories. New York: Orb Books, S. 1-4.

Coy, Wolfgang (1993): Der diskrete Takt der Maschinerie. In: Georg Christoph Tholen/ Michael Scholl/Martin Heller (Hrsg.): Zeitreise. Bilder, Maschinen, Strategien, Rätsel. Zürich: Stroemfeld/Roter Stern, S. 367-378.

Davis, Eric W. (2004): Teleportation Physics Study. Special Report. In: [https://www.fas.org/sgp/eprint/teleport.pdf], letzter Abruf: 30.04.2014.

Einstein, Albert/Rosen, Nathan (1935): The Principle Problem in the General Theory of Relativity. In: Physical Review Vol. 48, 1. Juli 1935, S. 73-77.

Ernst, Wolfgang: Medientheorie als Medienarchäologie – Einsichten in den technischen Vollzug. In: [http://www.medientheorien.hu-berlin.de/downloads/skripte/medientheorie1.pdf], letzter Abruf: 01.09.2014.

Farina, Mario V. (1970): Flowcharting: Getting Down to Basics. o. O.: Tektronix, Inc.

Frosh, Paul/Pinchevski, Amit (2009): Crisis-Readiness and Media Witnessing. In: The Communication Review, 12, 2009, S. 295-304.

Gilbreth, Frank B./Gilbreth, L. M. (1921): Process Charts. First Steps in Finding the One Best Way to do Work. New York: American Society of Mechanical Engineers. [https://archive.org/download/processchartsoogilb/processchartsoogilb.pdf], letzter Abruf: 30.04.2014.

Hodges, Don (2008): Bride of Kill Screen. The Journey to Find, Analyze, and Fix Ms. Pac-Man's Kill Screens. In: [http://donhodges.com/how_high_can_you_get3.htm], letzter Abruf: 30.04.2014.

Intersil (o. J.): 2114.4096 Bit (1024x4) NMOS Static RAM. In: [http://dk.toastednet.org/Vectrex/Datasheets/2114_-_RAM.pdf], letzter Abruf: 30.04.2014.

King, Greg (o. J.): Pacman Troubleshooting. In: [http://www.arcadegameover.com/pactrouble.html], letzter Abruf: 30.04.2014.

Kittler, Friedrich (2002): Computergrafik. Eine halbtechnische Einführung. In: Herta Wolf (Hrsg): Paradigma Fotografie. Fotokritik am Ende des fotografischen Zeitalters. Band 1. Frankfurt/M.: Suhrkamp, S. 178-194.

Krasnikov. S. V. (2011): Toward a Traversable Wormhole. In: Space Technology and Application International Forum, 2000. [http://arxiv.org/abs/gr-qc/0003092], letzter

Abruf: 30.04.2014.

Lawrence, Scott (o. J.): Programming for the Pac-Man or Pengo arcade platforms. In: [http://umlautllama.com/projects/pacdocs/], letzter Abruf: 30.04.2014.

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

Longridge, Mark (o. J.): Commented Disassembly of Pacman. In: [http://cubeman.org/arcade-source/pacman.asm], letzter Abruf: 30.04.2014.

Malone, Michael S. (1995): Der Mikroprozessor. Eine ungewöhnliche Biografie. Berlin u.a.: Springer.

Misner, Charles W./Wheeler, John A. (1957): Classical Physics as Geometry Gravitation, Electromagnetism, Unquantized Charge, and Mass as Properties of Curved Empty Space. In: Annals of Physics, Nr. 2 (1957), S. 525-603.

Naake, Frider (o. J.): Hier sind ein paar Sprüche. In: [http://www.hfk-bremen.de/en/profiles/n/frieder-nake], letzter Abruf: 30.04.2014.

N. N. (2000): PacMan PCB Modifications. In: [http://www.pinrepair.com/video/pacman.htm], letzter Abruf: 30.04.2014.

Pias, Claus (2002): Computer Spiel Welten. Wien: Diaphanes.

Rost, Rudolf (1966): Silicium als Halbleiter. Stuttgart: Berliner Union.

The Computer History Museum (o. J.): Mastering the Game. The History of Computer Chess. Kapitel: Middle Game: Computer Chess comes of Age. Unterkapitel: Fast and Efficient Searching. In: [http://www.computerhistory.org/chess/main.php?sec=thm-42eeab f470432&sel=thm-42f15c6fb49f2], letzter Abruf: 30.04.2014.

Vaas, Rüdiger (2013): Tunnel durch Raum und Zeit. Von Einstein zu Hawking – Schwarze Löcher, Zeitreisen und Überlichtgeschwindigkeit. Stuttgart: Kosmos.

Visser, Matt (1996): Lorentzian Wormholes: From Einstein to Hawking (AIP Series in Computational and Applied Mathematical Physics). New York u.a.: Springer.

Völz, Horst (2007): Handbuch der Speicherung von Information. Band 3: Geschichte und Zukunft elektronischer Medien. Aachen: Shaker.

Wheeler, John Archibald (1955): Geons. In: Physical Review, Vol. 97, No. 2, 15. Januar 1955, S. 511-536.

Wicks, Keith R. (1991): Fractals and Hyperspaces (Lecture Notes in Mathematics). New York u.a.: Springer.

Wolf, Fred A. (1990): Der Quantensprung ist keine Hexerei. Die Neue Physik für Einsteiger. Frankfurt/M.: Fischer.

Zaks, Rodneay (1987): Programmierung des Z80. Düsseldorf u.a.: Sybex.

Spiele

Archon (Electronic Arts 1983, Free Fall Associates)
Micro Chess (1976, Peter R. Jennings)
Ms. Pac-Man (Bally Midway/Namco 1982, Bally Midway/General Computer Corp.)
Spacewar! (1962, Steve Russel)