Harald Hillgärtner

Sauerbraten! Jawohl! Eine Game-Engine als Kollaborationsplattform

2009

https://doi.org/10.25969/mediarep/1901

Veröffentlichungsversion / published version
Sammelbandbeitrag / collection article

Empfohlene Zitierung / Suggested Citation:
Hillgärtner, Harald: Sauerbraten! Jawohl! Eine Game-Engine als Kollaborationsplattform. In: Matthias Bopp, Serjoscha Wiemer (Hg.): *Shooter. Eine multdisziplinäre Einführung.* Münster: LIT 2009 (Medien'welten. Braunschweiger Schriften zur Medienkultur), S. 267–284. DOI: https://doi.org/10.25969/mediarep/1901.

Harald Hillgärtner

Sauerbraten! Jawohl! Eine Game-Engine als Kollaborationsplattform

Abstract

Ausgehend von Marshall McLuhans These, dass das Medium selbst die Botschaft sei und nicht dessen Inhalte, zeichnete sich die Debatte um den Computer als Medium vor allem dadurch aus, dass zumindest im akademischen Diskurs den Nutzern relativ wenig Beachtung geschenkt wurde. Stattdessen wurde die Technik fokussiert, die Nutzer erschienen vor diesem hoch technischen Dispositiv lediglich als Anhängsel. Am Beispiel des Open-Source-Shooters SAUERBRATEN◂1 soll ein Perspektivenwechsel vorgenommen werden. Das Konzept dieses recht unbekannten Computerspiels besteht darin, dass die beiden Modi »Spielen« und »Editieren« ineinander verschränkt sind. Ohne jedwede Programmierkenntnisse und nur mit geringer Einarbeitungszeit kann der Spieler zum Co-Autor dieses Shooters werden. Deutlich soll werden, dass das lange in der Informatik bestimmende Konzept eines werkzeughaften Gebrauchs des Computers keinesfalls ausgedient hat, sondern auch und gerade im Computer als Medium wirkmächtig bleibt.

Der zu verfolgende Ansatz versucht dementsprechend, das Feld der Computerspiele vor dem Hintergrund einer zentralen medientheoretischen Position – der Verschränkung von Nutzer und Apparatur, von Mensch und Maschine – zu konturieren. Eine wissenschaftliche Auseinandersetzung mit diesem jüngsten Gegenstand der Kulturwissenschaft hat notwendig mit diesem Problem umzugehen: Computerspiele basieren auf einem hoch komplexen technisch-medialen Zusammenhang. Diese Diagnose soll jedoch keinesfalls dazu verführen, die Nutzerpraxen zugunsten der Technikanalyse zu vernachlässigen. Entscheidend ist vielmehr, beides in Beziehung zu setzen und als wechselseitige Adaptionsleistung zu verstehen.

Einleitung

»Um die Displays in den Cockpits auch unter den Bedingungen von Star Wars noch ablesen und bedienen zu können, kommt es auf Reaktionstempi im Millisekundenbereich an. Präsident Reagan hat nicht umsonst alle Freaks von Atari-Spielcomputern als zukünftige Bomberpiloten bewillkommnet.« (Kittler 1991, 256)

Friedrich Kittlers Steckenpferd, die innige Verwandtschaft von Militärtechnik und Medientechnik herauszustellen, ist weithin bekannt und er betreibt dies mit bemerkenswerter Kreativität. So einleuchtend dieser Zusammenhang in manchen Fällen sein mag, etwa beim Radio, dessen zivile Nutzung Kittler zufolge nichts anderes als ein »Mißbrauch von Heeresgerät« darstellt (vgl. Kittler 1991), ist dieser Sachverhalt bei anderen Techniken nicht ganz so unmittelbar einsichtig. Es dürfte nämlich weitaus weniger bekannt sein, dass auch die Apparatur zur Produktion von Kinofilmen ein Abkömmling des Revolvers ist:

»Alle Aufnahmegeräte für bewegte Bilder gehen auf Mareys chronophotographische Flinte zurück, die ihrerseits, wie der Name schon sagt, auf Gattlings Revolvergeschütz zurückgeht.« (Kittler 1992, 176f.)

Doch der Krieg ist nicht allein der Vater vieler Medientechnologien, sondern er infiziert ebenso deren zivile Nutzung. Der Unterschied zwischen Krieg und Frieden verschwimmt, wo die Unterhaltung, ob bewusst oder unbewusst, mitunter nichts anderem als einer allgemeinen Mobilisierung, einem Training der Massen für die zukünftigen Kriege, dient.
Wo läge der Konnex zwischen Unterhaltung und Mobilmachung näher als bei den Computerspielen und ihrem populärsten Genre, den Ego-Shootern? Lässt sich leugnen, was sich Kittler zu Folge bereits Ronald Reagan als Effekt der Computerspiele erhoffte? Gleichen nicht die gegenwärtigen Bilder militärischer Operationen im Irak teilweise auf erschreckende Weise den Bildern, die man bereits aus neueren Kriegssimulationsspielen gewohnt ist?
Es scheint, als könne sich Kittler bestätigt sehen. Seine ungemilderte Negativität, die ihn – und sei es nur in diesem einen Punkt – ungewollt mit der Kritischen Theorie verbindet, war angebracht: Das unter der Schirmherrschaft des amerikanischen Verteidigungsministeriums entwickelte und distribuierte Spiel America´s Army (MOVES Institute/United States Army 2002) vereint Unterhaltung und Mobilisierung mit einer erstaunlichen Selbstverständlichkeit.◂2
Doch man sollte sich von solch allzu offenkundigen Zusammenhängen nicht in die Irre führen lassen. Das Projekt einer technikzentrierten Medienwissenschaft versteht sich nicht als Ideologiekritik im herkömmlichen Sinne. Nicht die

gesellschaftlichen Verwendungsweisen einer Technologie werden analysiert und kritisiert, das Erkenntnisinteresse setzt bereits vor dieser sozialen Ebene an. Es ist die Technik selbst, in die ihre Verwendungsweisen als historischer Index eingeschrieben sind. Jedwedes Computerspiel ist in diesem Sinne als ein Abkömmling der Radarbildschirme und ihr Interface als Fortführung der Steuerknüppel in den Kampfflugzeugen zu begreifen (vgl. Halbach 1994). So dienen eben auch solche ›harmlosen‹ Spiele wie Extreme Tux Racer ◂3, bei dem ein Pinguin bäuchlings eine schneebedeckte Piste hinunterrast, um dabei möglichst viele Heringe aufzuschnappen, ebenfalls dem Training der Reaktionsgeschwindigkeit und bilden damit letztlich potenziellen Nachwuchs für Kampfbomber aus. Doch auch wenn sich ein solcher Ansatz insbesondere in Hinsicht auf Ego-Shooter und Kriegssimulationen geradezu aufdrängt: Die gleiche Technik muss nicht notwendig im Dienste einer Mobilmachung stehen. Der Computer und mit ihm die Computerspiele mögen eine missbräuchliche Verwendung von Heeresgerät sein, doch damit ist das Phänomen der Shooter beileibe noch nicht abgehandelt.

Im Folgenden soll es also um eine durchaus zivile Nutzung gehen, die – so die Hoffnung – ein anderes Licht auf die Shooter wirft. Gemeint ist ihre Verwendung als Plattformen kollaborativer Produktion, in der die Spieler eben nicht schlechterdings für den Kriegsdienst herangebildet werden, sondern sich an dem Prozess des Um-, Fort- und Weiterschreibens des Spieles beteiligen.

Game-Engines

Der Erfolg der von IBM in den 1980ern mit hohem Aufwand betriebenen Entwicklung des PCs liegt nicht zuletzt in der frühen Designentscheidung, ihn als modulares System zu konzipieren. Die einzelnen Komponenten können unabhängig voneinander von unterschiedlichen Hardwareherstellern entwickelt werden, wobei sie über standardisierte Schnittstellen miteinander kommunizieren. Ganz ähnlich funktioniert auch Software im Wesentlichen modular. Insbesondere auf grafischen Benutzeroberflächen (GUI) basierende Systeme stellen eine Reihe von Bibliotheken zur Verfügung, in der bestimmte Prozeduren und Funktionalitäten über eine definierte Schnittstelle (API) bereitgestellt werden. In den Bibliotheken finden sich zudem Steuerelemente für die Benutzeroberflächen (Widgets), die nicht allein den Programmieraufwand bei der Erstellung neuer Applikationen minimieren, sondern eben auch einen konsistenten Stil, ein einheitliches ›Look‘ n‘ Feel‹ ermöglichen.

Im Bereich der Computerspiele übernimmt die Game-Engine die Aufgabe einer solchen Bibliothek. Die Game-Engine gibt den Rahmen vor, auf dem mitunter ganz und gar unterschiedliche Spiele entwickelt werden können. Die Kernaufgabe einer Game-Engine ist hierbei erwartungsgemäß das Grafiksystem. Dieses beinhaltet Funktionen etwa zum Verwalten und zur Darstellung von Texturen oder 3D-Objekten, wichtig sind aber auch die Fähigkeiten zur Berechnung der Beleuchtung (Shader), die wesentlich zum optischen Realismus eines Spieles beitragen und solche Effekte wie Nebel, Wasser oder Feuer ermöglichen. Die Simulierung physikalischen Verhaltens wird gleichfalls von der Game-Engine übernommen bzw. werden sogenannte Physik-Engines in jüngerer Zeit als eigene Sub- oder Co-Systeme implementiert. Grob formuliert stellt diese sicher, dass den Gesetzen des Newton-Raums Geltung verschafft wird, wobei jedoch nur vordergründig größtmöglicher Realismus angestrebt wird, tatsächlich aber lediglich die physikalische Intuition der Spielerin adressiert wird (Poole 2000, 59ff.). Größeren Raum nimmt ebenfalls die KI einer Game-Engine ein, die insbesondere die Fähigkeiten etwaiger Gegner in einem Shooter bestimmt, die nicht von einem menschlichen Spieler gesteuert werden. Darüber hinaus beinhalten Game-Engines noch gegebenenfalls die Netzwerkfunktionalität eines Spieles, das Soundsystem, die Spielsteuerung und das Datenmanagement, mit dessen Hilfe sich etwa Spielstände abspeichern lassen. Im Bereich des PCs sind die Funktionalitäten einer Game-Engine sowohl von der Hardware einer Konfiguration, insbesondere von der Grafikkarte, als auch von deren Abstraktionsschichten wie OpenGL, DirectX oder SDL◂4 abhängig.

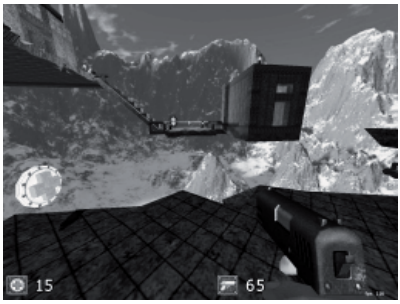
Die Game-Engine stellt hiermit die Infrastruktur bereit, auf der das eigentliche Gameplay stattfindet. Dieses ist jedoch nicht mehr die Aufgabe der Programmiererinnen, sondern bereits die von (Spiele-)Designern, wobei hier die Übergänge sicherlich fließend sind. Um den Zugriff auf ihre Infrastruktur zu ermöglichen, verfügen Game-Engines in aller Regel über Skript-Sprachen, die auch für Nichtinformatiker einfach zu erlernen sind, und mit deren Hilfe die Spielabläufe geschrieben werden können. Deutlich ist jedenfalls, dass Computerspiele nicht notwendig ein vorfabriziertes, unveränderliches und in eine Klarsichtfolie eingeschweißtes ›Shrink-Wrap‹›-Produkt sind, sondern in manchen Fällen auch und gerade eine Art ›Computerbetriebssystem‹, auf dessen Basis Modifikationen vorgenommen werden können und sollen.

Klar ist, dass das Lizenzieren von Game-Engines ein recht ertragreiches Geschäft darstellt. Hierbei finden sich auf dem Markt sowohl kommerzielle als auch eine Reihe von Game-Engines auf Open-Source-Basis, die teilweise unter der GPL lizenziert sind. Glücklicherweise wird hier der Markt gerade nicht, wie etwa bei Computerbetriebssystemen, von wenigen Anbietern dominiert. Gera-

dezu erstaunlich ist es, wie lebendig sich die Entwicklung im Bereich der Open-Source-Game-Engines inzwischen darstellt.◂5 Dies steht gar in einem ebenso merkwürdigen Kontrast zu dem Umstand, dass es doch im Grunde recht wenig eigenständige Computerspiele gibt, die unter den unterschiedlichen Unix-Derivaten lauffähig sind. Zwar gibt es mit ›SDL‹ bereits seit einiger Zeit ein – wenn auch sehr eingeschränktes – Pendant zu Microsofts »DirectX«, jedoch bleibt es dabei: Wer gerne auf seinem PC aufwendige Computerspiele spielen möchte, kommt an einer Windows-Installation nicht vorbei. Selbst die wenigen Open-Source-Spiele, die es gibt, stehen dabei wie selbstverständlich in einer Windows-Portierung zur Verfügung. Es bleibt somit bei der alten Einteilung: Die Entwicklerinnen von Software arbeiten gerne mit Unix-Derivaten, die Anwender hingegen mit Microsoft-Produkten.

Dass das Entwickeln von Game-Engines nicht nur im Open-Source-Bereich so prominent geworden ist, hängt offenbar nicht zuletzt mit dem Entschluss der bekannten Computerspiele-Schmiede id Software zusammen, die älteren erfolgreichen Wolfenstein-, Quake- und Doom-Engines zunächst im Quellcode zu veröffentlichen und später unter der General Public License (GPL) zu lizenzieren, sie also zu freier Software werden zu lassen. Nicht allein die künstlerische Bearbeitung von Computerspielen wurde durch diesen Entschluss massiv befördert, sondern eben auch die Entwicklung von Computerspielen als Freizeitbeschäftigung. So listet etwa ›Doomworld‹ nahezu einhundert verschiedene Portierungen der Doom-Engine auf die unterschiedlichsten Architekturen◂6, wobei sicherlich die Versionen für verschiedene Handhelds oder gar für Kodak-Fotoapparate◂7 die bemerkenswertesten sind.

Doch bleibt es bei diesen Bearbeitungen nicht dabei, den ursprünglichen Quellcode von id Software zu verbessern, neue Features zu implementieren oder den Code an alle denkbaren Geräte anzupassen, deren Chips irgend leistungsfähig genug sind, um mit ihnen Doom (id Software/id Software1993) spielen zu können. Vielmehr entstehen nach und nach Ego-Shooter, die zwar etwa auf den verschiedenen Quake-Engines basieren, jedoch in Bezug auf Ästhetik, Funktionalität und Gameplay – statt einfacher Modifikationen – ganz und gar eigenständige Entwicklungen darstellen. So verfügt das »Retro-Sci-Fi« Spiel Alien Arena◂8 inzwischen über eine breite Community mit einer Reihe von aktiven Servern, Tremulous ◂9 beeindruckt durch sein ausgefeiltes Gameplay, Nexuiz◂10 hingegen durch Grafik und Geschwindigkeit, ästhetisch äußerst interessant ist zudem der comicähnliche Stil von War§ow◂11, dessen Gameplay darüber hinaus durch die verschiedenen möglichen Bewegungsabläufe als »sportlich« bezeichnet werden kann.



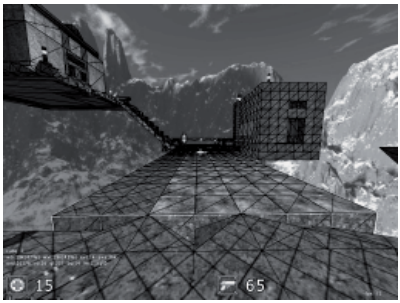


Abb. 1 und 2: In-Game Editing: Sauerbraten integriert den Spiele- und den Editor-Modus nahtlos. Ein unüberwindbarer Abgrund wird innerhalb von Sekunden begehbar.

Neben den Quake- oder Doom-Derivaten existieren im Open-Source-Bereich weiterhin Projekte, die nichts mit dem Code der Firma id Software zu tun haben, und die auf einer ganz anderen Game-Engine basieren, deren Ästhetik sich dementsprechend mehr oder weniger von Doom oder Quake (id Software/Activision 1996) unterscheidet. Als Beispiel sei hier Crystalcore genannt, das auf der Crystal-Space-Engine◂12 aufbaut. Eine eigenständige Entwicklung ist ebenfalls die CUBE-Engine◂11 des belgischen Entwicklers Wouter van Oortmerssen. Zwar lässt sich Cube auf den ersten Blick nicht anders als Quake-ähnlich bezeichnen, dennoch unterscheiden sich Cube und sein Nachfolger Sauerbraten auf einer anderen Ebene signifikant von anderen Ego-Shootern.

Sauerbraten

Ausschlaggebend für das Design der Sauerbraten-Engine war Oortmerssen zu Folge »Einfachheit«. Statt auf komplexe, möglichst realitätsgetreue Grafikfunktionalitäten, also auf »Eye Candy« zu setzen, kommen bei den *maps* in Sauerbraten einfach zu erzeugende ›Octrees‹ zum Einsatz. Hierbei handelt es sich um Kuben, die sich in sämtliche Richtungen um weitere Kuben ergänzen lassen. Diese baumartig aneinandergeknüpften Würfel lassen sich zudem jeweils in acht kleinere Würfel unterteilen, aus denen wiederum acht Würfel erstellt werden können usw.◂14 Sobald man den Editor-Modus mittels eines einfachen Tastendrucks einschaltet, erscheint einer der bereits vorhandenen Kuben im Fadenkreuz des Spielers umrandet. Diesen kann man nun mit der Maus markieren und mittels des Mausrades editieren. So lassen sich in aller Schnelle neue Wände, Treppen oder ein Übergang über einen Abgrund einfügen. Ebenso umstandslos lässt sich ein neuer Durchgang durch eine bestehende Wand erzeugen: Editor-Modus einschalten, ein Quadrat aus der Wand auswählen, am Mausrad drehen, fertig.

Eine neue *map* erstellt man mit dem Befehl ›/newmap‹ in der Kommandozeile, wobei hier über einen Parameter die Größe der in der Luft schwebenden Grund-

fläche mit übergeben werden kann. Die Grundfläche besteht – wie nicht anders zu erwarten – aus Kuben, auf denen nun mittels der Maus die gewünschten Gebäude erstellt werden können. Die ersten eigenen vier Wände entstehen so innerhalb von ein paar Sekunden. Sukzessive lassen sich so auch komplexe Geometrien erzeugen. Mittels weiterer Eingaben innerhalb des Editor-Modus lassen sich die Kanten der geometrischen Formen abschrägen oder mit bereits vorab definierten Texturen belegen. Aus einem gesonderten Menü heraus stehen eine Reihe von ›Entities‹ zur Verfügung. Hierbei handelt es sich etwa um verschiedene ›Monster‹ mit je unterschiedlichen Fähigkeiten. Weitere Entities sind Munition, Rüstungen, Bäume oder Blumen, Lichtquellen, Respawn-Punkte, Teleporter, ›Health-Items‹ etc. Das Inventar stellt alles bereit, was die Sauerbraten-Engine an Attraktionen vorsieht.

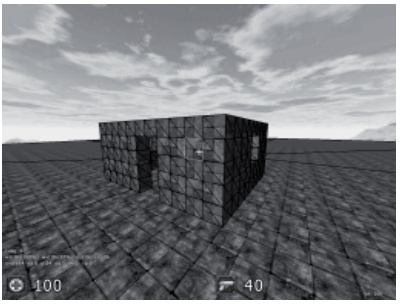


Abb. 3: In kürzester Zeit zu den ersten eigenen vier Wänden ...

Was nun wie eine Spielerei klingt, ist auch eine. *Map*-Editing verlangt keinerlei Programmierkenntnisse und lediglich eine Einarbeitungszeit von nur wenigen Minuten: Nicht zuletzt umfasst das den Sourcen beigelegte *Readme* zum Punkto Editieren, würde man alle Abbildungen weglassen, lediglich zwei Seiten, aus denen zunächst nicht mehr zu erfahren ist als eben dargelegt. Mit der Maus klicken, neue Wände durch das Ziehen der Maus erstellen und am Scrollrad drehen, um die Kuben zu skalieren. Gegenstände einfügen oder löschen. Fertig. Dies alles, wohlgemerkt, auch während eines laufenden Spiels! Sollten einem in einer Singleplayer-*Map* die schiere Menge der Angreifer zu viel werden oder sollten einem die Sprünge über einen Abgrund zu gewagt sein: Mit einem Druck auf die E-Taste lassen sich die Probleme aus der Welt schaffen.

Freilich lässt sich die Komplexität des Editierens bei dem Erstellen einer neuen *map* fast beliebig steigern. Vor welche Schwierigkeiten man sich gestellt sieht, vermittelt ein etwas umfangreicheres Tutorial (vgl. Bekel 2005) auf der dem Spiel Cube und seinen Derivaten gewidmeten Webseite ›Quadropolis‹◂15. Der Spielspaß – und dies ist sicherlich kein Geheimnis – hängt wesentlich von einer gelungenen *map* ab. So ergibt es keinen Sinn, einen Spieler mit allzu viel Munition insbesondere für das Maschinengewehr auszustatten, ebenso wie es durchaus frustrierend sein kann, wenn zu viele mächtige Monster auf den Spieler einstürmen. Eine besondere Herausforderung besteht jedoch darin, die allzu statische Grundordnung der Kuben aufzubrechen und etwa gebogene Formen zu erzeugen. Will man darüber hinaus auch eigene 3D-Objekte erstel-

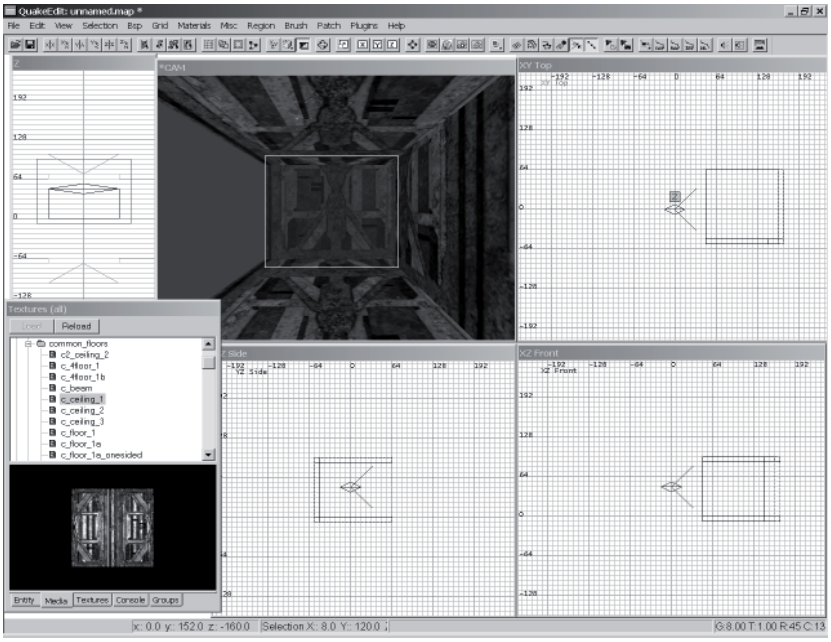


Abb. 4: Der Level-Editor von QUAKE 4 ist eine eigenständige und hoch komplexe Applikation.

len, also Gegner oder anderes Inventar, ist man allerdings auf die Hilfe anderer Software angewiesen, der eingebaute Editor hilft hier nicht weiter. Dieser beschränkt sich weitgehend auf die Grundfunktionalitäten: Landschaften und Architekturen erzeugen, Entities einfügen und für die richtige Beleuchtung sorgen.

QUAKE

Ganz anders sieht dies bei dem mitgelieferten Editor des populären Ego-Shooters Quake aus, der mittlerweile in seiner vierten Version erschienen ist. Hier findet das Editieren nicht innerhalb eines Spieles statt, sondern über das Starten eines eigenen Editors mit einer eigenen Benutzeroberfläche. Diese stellt eine Reihe von Fenstern zur Verfügung, die von vornherein ihren Zweck verdeutlichen: die Modellierung von Architektur(en). So gibt es ein ›Kamera‹-Fenster, das einen Eindruck von den späteren Räumen erlaubt. Andere Fenster stellen hingegen Abstraktionen der Architektur zur Verfügung, die deutlich an

technische Zeichnungen erinnert und jeweils eine Ansicht des Grundrisses, des Aufrisses und des Seitenrisses sehen lassen. Mit der Maus können nun in diesen schematischen Ansichten neue Objekte erstellt und deren Proportionen verändert werden, über Menüs und Submenüs steht allerlei Inventar zur Verfügung, über Shortcuts lassen sich verschiedene Funktionen adressieren, um etwa Wände zu kopieren oder Texturen zu skalieren. Es ist hier ein außerordentlich genaues Arbeiten möglich, sodass dem Feinschliff der *brushes*, also der Objekte, größte Aufmerksamkeit geschenkt werden kann. Das Arbeiten mit dem Quake-Editor ähnelt der Softwareentwicklung mittels einer grafischen Umgebung. Geradezu folgerichtig bietet id Software ein Quake4SDK an, also ein Software Development Kit für Quake 4◂**16**. SDKs beinhalten alle notwendigen Tools und Utilities sowie die entsprechenden Dokumentationen, um Software zu entwickeln. Ebenso wie bei dem Großteil gängiger Software findet sich hier eine Unterscheidung zwischen dem Quellcode und den Binaries wieder: Um eine spielbare Version einer neuen *map* zu erhalten, muss diese zunächst kompiliert werden.

Der Unterschied zwischen Sauerbraten und Quake lässt sich ähnlich dem Unterschied zwischen skriptbasierten Programmiersprachen und kompilierten Programmiersprachen fassen. Bei Ersteren steht der Quellcode jederzeit zur Verfügung, denn er wird erst zur Laufzeit interpretiert, bei Letzteren muss der Quellcode hingegen unabhängig von den *maps* mitgeliefert werden, damit die Nutzer Veränderungen vornehmen können. Dieser Unterschied findet sich auch im Lizenzmodell wieder. Wohingegen id Software ihre jeweils älteren Game-Engines als freie Software veröffentlichen, sind die Inhalte des Spiels, die *maps*, 3D-Modelle, Texturen, Soundeffekte etc., also sein »Media«, dennoch nicht zum freien Download verfügbar. Möglich wird hierdurch zwar, auf Basis des Quellcodes Modifikationen der Engine vorzunehmen oder die originale Software durch eigene Inhalte zu einem voll funktionsfähigen Spiel auszubauen, jedoch lässt sich nicht allein mit der freigegebenen Game-Engine Quake oder Doom spielen. Hierfür bedarf es nach wie vor einer proprietären Lizenz.

Kurzum: Der Mangel an Komplexität bzw. »Eye Candy« bei Sauerbraten gegenüber dem hier gewählten Beispiel Quake 4 resultiert aus der konsequenten Umsetzung des Konzeptes einer nahtlosen Integration der beiden Modi »Spielen« und »Editieren«. Beide sind – wie gesagt – bei Sauerbraten nur einen einzigen Tastendruck voneinander entfernt. Vergleichbar ist dieses integrative Konzept vor allem mit dem weitgehend unbekannten Webbrowser des W3-Konsortiums namens ›Amaya‹. Obschon Amaya als ›Referenzbrowser‹ zu gelten hat, da er als einziger sämtliche Webstandards implementiert, taucht er doch in kaum einer Browserstatistik auf, da ihn kaum einer verwendet. Amayas ›Pro-

blem‹ ist sicherlich, dass viele der Optimierungen, die von den Webdesignern eigens für andere (marktbeherrschende) Browser ersonnen wurden, von Amaya nicht berücksichtigt werden. Auch hier ist gewissermaßen das ›Eye Candy‹ ein Problem. Zudem unterstützt Amaya viele der proprietären Erweiterungen nicht und lässt damit einige Webseiten schlicht dysfunktional werden. Kurz: Wer sich Ärger ersparen möchte, sollte Amaya nicht einsetzen. Wirklich sinnvoll ist dieser Browser im Grunde nur für Webdesigner, die hiermit ihre Seiten auf Standardkonformität prüfen könnten, vorausgesetzt freilich, sie haben daran überhaupt ein Interesse.

Tim Berners-Lee – der ›Erfinder‹ des WWW – hatte jedoch »einen Traum für das Web«, in diesem Traum

> »wird das Web zu einem wesentlich leistungsfähigeren Werkzeug für die Zusammenarbeit von Menschen. Ich habe mir den Informationsraum immer als etwas vorgestellt, auf das jeder sofortigen und intuitiven Zugriff hat, und das nicht nur durchsucht, sondern in dem etwas erstellt werden kann« (Berners-Lee 1999, 229).

Lesen und Schreiben gehören also für Berners-Lee unbedingt zusammen und genau dieser Aspekt findet sich in Amaya wieder. Jede hiermit aufgerufene Webseite lässt sich unmittelbar editieren, einfach, indem man mit der Maus in den angezeigten Text klickt und losschreibt. Dass dieses Modell nicht ohne Weiteres praktikabel ist, liegt auf der Hand. So lässt sich eine veränderte Webseite nicht veröffentlichen, solange man keine Schreibrechte auf dem entsprechenden Webspace hat. Zudem sind die meisten Webseiten nicht dazu gedacht, von den Besuchern verändert zu werden. Jedoch ist es der Impetus, eine nahtlose Integration von Distribution und Partizipation zu ermöglichen, der Amaya auszeichnet: »This follows the original vision of the Web as a space for *collaboration* and not just a one-way publishing medium.« (Vatton 2006, Hervorhebung vom Verf.) Diese »Vision« schlägt sich übrigens auch darin nieder, dass das Web gewissermaßen ›Open Source‹ ist. Der HTML-Quelltext einer Webseite ist jederzeit einsehbar und dementsprechend lässt sich das Layout einer Webseite kopieren und den eigenen Gegebenheiten anpassen. Was dies bedeutet, lässt sich an der recht verbreiteten Adobe-Flash-Technologie (früher Macromedia-Flash) für Vektorgrafiken ersehen. Im Gegensatz zum vergleichbaren, aber unbeliebten SVG-Standard des W3-Konsortiums, sind Flash-Animationen nicht durch die Nutzer adaptierbar, da sie lediglich in einem maschinenlesbaren Binary-Format zur Verfügung stehen.

Was Amaya für das Web ist, wäre Sauerbraten für den Bereich der Shooter: Weitgehend unbekannt, aber in der Umsetzung eines kollaborativen Konzeptes konsequent. In diesem Sinne ließe sich Sauerbraten ebenfalls als »col-

laboration space« kennzeichnen, ist doch auch hier der Editor-Modus nahtlos integriert, vor allem aber lassen sich die *maps* über ein Netzwerk mittels des »coop edit«-Modus durch mehrere Mitspieler gleichzeitig umbauen. Mit diesem Merkmal ist Sauerbraten übrigens eigenen Angaben zufolge einzigartig, und die Technologie der Game-Engines, ihre Netzwerkfunktionalität, öffnet sich damit über die kollektive Rezeption hinaus zur kollektiven Produktion.

Skripte

Der Begriff ›Skripte‹ steht im Bereich des Computers für – meist kürzere – Texte, die in einer der verschiedenen Skript-Sprachen, wie etwa Perl, Python oder Ruby verfasst sind. Im Gegensatz zu anderen Programmiersprachen werden Skripte erst zur Laufzeit von einem Interpreter in Maschinencode übersetzt. Häufig werden Skripte zum Automatisieren immer wiederkehrender Aufgaben am Computer eingesetzt, wie insbesondere bei Bash-Skripten◂**17**, mit komplexeren Skriptsprachen können aber auch ganz eigenständige Anwendungen programmiert werden. Game-Engines beinhalten ebenfalls in aller Regel Skripting-Funktionalitäten, die beispielsweise das Erstellen und Editieren von *maps* erleichtern oder die Administration eines Spiele-Servers. Bei Skripten auf dem Computer geht es also um das Programmieren mithilfe einer Programmiersprache und damit um das Verfassen eines Textes im weiteren Sinne. Im Zusammenhang mit den Game-Engines soll jedoch der Versuch unternommen werden, den Begriff des Skriptes in einem anderen Sinne zu verwenden.

Vilém Flusser, der in seinem Essay *Die Schrift. Hat Schreiben Zukunft?* das Kunststück unternimmt, mit einem geschriebenen Werk die Schrift zu verabschieden oder doch zumindest über die Schrift hinauszuweisen, misst den »Script writers«, eine Formulierung, die – wie Flusser lakonisch bemerkt – etymologisch eigentlich »Schreibritzer« (Flusser 2002, 128) meint, einen hohen Stellenwert zu. Sie markieren bereits den Übergang von der Alphabetkultur hin zu der von ihm prognostizierten Zukunft. Diese erst in ihren Grundzügen absehbare Zukunft werde von einer Vorherrschaft der Bilder, genauer: der technischen Bilder, geprägt sein. Die Bezeichnung »Skript« meint nämlich in einem ganz anderen Kontext als dem des Computers, bei Radio, Film und Fernsehen, eine Vorlage bzw. »Vorschrift«, auf deren Basis Filme gedreht oder Sendungen produziert werden.◂**18** Es sind diese »Script writer«, die Flusser hier adressiert. Sie stehen bereits an der Schwelle zwischen der alphabetischen und der nachalphabetischen Kultur und seien dementsprechend lediglich ein Übergangsphä-

nomen. Sie schreiben als Autoren nicht an einen Leser, sondern an den Apparat der »Film-, Fernseh- und Hörfunkproduzenten«, damit diese aus ihren Texten Bilder erzeugen, wobei Flusser zufolge die Radioskripte, die sich an »bilderlose Sprecher« richten, zuallererst verschwinden werden.
Für Flusser stellt ein Skript etwas Hybrides dar:

»Halb ist es noch ein Text für ein aufzuführendes Drama und als solcher ein Nachkomme des Sophokles, halb ist es bereits Apparatprogrammierung und als solches Vorfahre der von künstlichen Intelligenzen automatisch kalkulierten Programme.« (ebd.)

Es ist diese textuelle Altlast, die es zu überwinden gelte, denn die Skripte seien tatsächlich keine Texte mehr, sondern »Prätexte«: »Das Alphabet ist bei ihnen zu einem Hilfscode geworden. Seine Absicht ist, Bilder zu machen.« (ebd., 130) Diese Übergangserscheinung der Skripte gebe es auch nur, da noch »überall alphabetisch geeichte Geräte herumliegen«. (ebd., 131) Der Text ist die »Pferdekutsche«, die noch nicht den bereits erfundenen Autos weichen will, deren Ende aber bereits abzusehen ist:

»Sehr bald wird man nicht mehr in Pferdekutschen den Bildern entgegenfahren müssen und die Skripte werden funktioneller codierten Bildvorschriften weichen. [...] Skripte sind der Schwanengesang der Texte.« (ebd.)

Ob nun die Gegenwart tatsächlich in einem *Universum der technischen Bilder* – so der Titel eines anderen, sehr bekannten Essays Flussers – münden wird, und wie sich eine solche schriftlose Kultur darstellt, sei an dieser Stelle dahingestellt. Interessant ist, wie Flussers Texte von einem kaum zu überbietenden affirmativen Gestus gegenüber den (technischen) Bildern getragen werden. Dies ist ein Gesichtspunkt, der, wollte man diesen Gestus verstehen, wesentlich breiter diskutiert werden müsste, als es in diesem Rahmen möglich ist. Trotzdem: Die Technik des *Map*-Editings ließe sich bereits als Überwindung des Skript-Schreibens im flusserschen Sinne auffassen. Man schreibt zwar noch in einem übertragenen Sinne Skripte, indem man den zukünftigen Spielern einer *map* Handlungsräume und auch Handlungsverläufe vorschreibt, man ist also in ebenso übertragenem Sinne nach wie vor »Dramaturg«, jedoch bedient man sich nicht der »Pferdekutschen« des alphabetischen Textes, sondern schreibt bereits auf Basis von etwas Neuem, der Game-Engine.
Ein bemerkenswertes Phänomen nicht nur der gegenwärtigen Film- und Fernsehkultur ist in diesem Zusammenhang die ›Fan-Fiction‹. Anhänger etwa der Serie The X-Files (USA 1993–2002, Chris Carter) publizieren ihre selbst verfassten Plots auf entsprechenden Webseiten◂**19**, doch es bleibt meist bei dieser ›virtuellen‹ Teilhabe an der Serienproduktion. Im Gegensatz hierzu liegt

ein besonderer Reiz bei Computerspielen wie Quake, Doom, Counterstrike und nicht zuletzt Sauerbraten darin, dass sie die Spieler eben auch zu Scriptwritern werden lassen, da die Editoren den Ego-Shootern bereits wie selbstverständlich beiliegen oder passend zu den Spielen aus dem Internet als Open-Source-Software herunterladbar sind. Dass nun Quake oder Doom so populär sind, liegt nicht allein am Gameplay, sondern an der Offenheit für Modifikationen dieser Spiele. Auf Basis quelloffener Game-Engines lassen sich noch weitgehender ganz eigenständige Spiele entwickeln, und auch Counter-Strike (Minh Le & Jess Cliffe/EA Games 2001) ist nichts anderes als die Weiterentwicklung eines anderen Spieles. Ego-Shooter sind somit nicht zuletzt auch »Mitschreib-Projekte« und mit Eisenstern◂20 steht nun auch eine Rollenspiel-Version von Sauerbraten in den Startlöchern, in der dem Genre entsprechend eine ›Entwicklung‹ der Spielfiguren möglich werden soll. Aus Cube wurde Sauerbraten und aus Sauerbraten wird Eisenstern.

Im Grunde gilt nach wie vor, was Flusser über die Drehbuchschreiber feststellt: Sie vollbringen ihr Werk im Verborgenen, ihre »Zirkusnummer« ist »kein öffentliches Spektakel« und müsste »erst dem Output der Medien« (ebd., 128) entnommen werden. Die Stars der Filme sind die Regisseure, mehr noch die Schauspieler; der unangefochtene Star der Computerspiele ist hingegen die Technik, teilweise bereits die Spieler. Die Autoren der Spiele, der *mods* und der *maps* treten, wie die Drehbuchschreiber, in den Hintergrund. Ihre Leistung gilt es, dem sichtbaren Output der Spiele zu entlesen. Diese Sichtbarkeit wird gewöhnlich der Technik zugeschrieben, den realistischen 3D-Effekten, dem möglichst realitätsgetreuen Licht- und Schattenwurf, den atmosphärischen Effekten wie Nebel oder Regen, den detailgetreuen Auflösungen der Texturen etc. Zweifellos: Mit dem Einsatz des bestmöglichen ›Eye Candy‹ steht und fällt die Popularität eines Spieles. Dahinter verborgen sind jedoch diejenigen, die auf Basis dieser Technik überhaupt erst für das Sichtbare und Spielbare sorgen, sie treten in Interaktion mit der Technik und versuchen, diese in ihren Fähigkeiten auszureizen.

Medientechnik – Do It Yourself?

Von hier aus wäre nun erneut die Ausgangsfrage nach dem Zusammenhang von Medientechnik und ihren sozialen Gebrauchsweisen in den Blick zu nehmen. Spätestens seit McLuhan gelten Medien nicht mehr als neutrale Mittler, sondern stehen in Verdacht, die Art und Weise des Denkens, des In-der-Welt-Sein, entscheidend zu beeinflussen, gar zu determinieren. Der Buchdruck ver-

einheitlicht nicht bloß die Nationalsprachen, sondern den ganzen Menschen. Der Gutenberg-Mensch ist ein uniformer Mensch. (vgl. McLuhan 1995) Der medientheoretisch interessierte Blick richtet sich in der Nachfolge McLuhans auf die Technik(en) der Medien, weniger auf deren Inhalte. Sie sind das Dispositiv, der dem Bewusstsein verborgene nichtsprachliche Anteil der Diskurse, der freigelegt werden muss, um den Einfluss der Medien auf die Subjekte erkennen zu können. (vgl. Winkler 1994) In der Radikalisierung dieses Gedankens sind die Medien nicht mehr Extensionen des Menschen – als die sie McLuhan noch bezeichnet hat –, sondern die Subjekte erscheinen als Extensionen ihrer Technik:

»Die Unterstellung, der Mensch sei Herr oder, wo nicht, der Sklave der Technik, ist ein langlebiges Axiom. Gestritten wird [...] über die Rollenverteilung, nicht aber über das Schema von Herr und Knecht, das unser anthropologische Vertrauen und Mißtrauen gegenüber der Technik ausmacht.« (Tholen 2002, 190)

In einer herkömmlichen Perspektive sind die Medien nichts weiter als Werkzeuge der Menschen. Sie sind transparent in ihrem Gebrauch, sie stehen im Wortsinne zur Verfügung. Trägt man nun der poietischen, der welterzeugenden Leistung der Medien Rechnung, dann verkehrt sich diese Perspektive. Mit einem Mal gibt es keine Meisterschaft über die Medien mehr, vielmehr erscheinen die Subjekte als Produkte der vormaligen ›Werkzeuge‹. Dieses Schema nun wäre zu überwinden, so das Plädoyer Georg Christoph Tholens, wobei ebenso wenig von einem einfachen Werkzeugbegriff der Medien auszugehen sei wie von einer reinen Selbsttätigkeit, einer ungetrübten Autonomie der technischen Artefakte, die – so Tholens Analyse – nichts anderes als eine Anthropologisierung der Maschine mit negativem Vorzeichen darstellt. Insbesondere in Hinsicht auf die Debatte um den Computer als Medium hatte diese Anthropologisierung geradezu Konjunktur. Dies muss nicht erstaunen, scheint doch der Computer als universelle Maschine besonders dazu geeignet, das menschliche Privileg des Denkens infrage zu stellen und hat doch bereits der ›Erfinder‹ des Prinzips der universellen Maschine mit dem ›Turing-Test‹ genau diese Fragestellung aufgeworfen. Lässt sich nicht mehr unterscheiden, ob der Partner in einer Kommunikation ein Mensch oder eine Maschine ist, so muss man der Maschine so etwas wie Intelligenz beimessen, auch wenn sie im strengen Sinne nicht ›intelligent‹ ist (vgl. Heintz 1993, 261ff.). Die autonome Maschine, die inzwischen zum Medium avanciert ist, wäre nun der letztgültige Schritt, die eine Rede von den Medien als Werkzeuge des Menschen obsolet werden lässt.
Trotzdem: Eine werkzeughafte Verwendung des Computers wäre hiermit nicht ausgeschlossen. Zwar kann kein naiver Werkzeugbegriff im Sinne einer jederzeit gegebenen Transparenz des technischen Artefakts in Anschlag gebracht

werden, da die zunehmend komplexer werdende Konfiguration aus Hard- und Software, mit der wir gewohnt sind umzugehen, eben diese Transparenz nicht zulässt. Die Programmierbarkeit der Maschine erlaubt es jedoch, den Umgang mit dem Computer werkzeughaft auszugestalten und das entsprechende Konzept war und ist eine der wesentlichen Leitmetaphern in der Anwendungsentwicklung (vgl. Schelhowe 1997, 92ff.). Klar ist, dass diese ›Software Tools‹ bloß nachgebildete, gewissermaßen ›verdoppelte‹ Werkzeuge sind (vgl. Coy 1995), dennoch ist mit ihnen ein einübender, ein verlässlicher Umgang möglich. Der Computer ist in diesem Sinne ebenso Distributionsmedium wie er – als programmierbare Maschine – Produktionsmedium ist.

Wie stellt sich nun dieser Sachverhalt am Beispiel der Computerspiele dar? Zunächst scheint sich diese Frage quasi von selbst zu beantworten: Wie bei kaum einer anderen Software sind Computerspiele interaktiv. Die Maschine reagiert unmittelbar auf die Tastatureingaben, die Spielerin hingegen ebenso unmittelbar auf das Geschehen auf dem Bildschirm. Beide sind in einer Feedbackschleife eingebunden, die ein Herr-Knecht-Schema schlicht aufzuheben scheint. Die gerade in Ego-Shootern geforderten Reiz-Reaktions-Geschwindigkeiten lassen den Spieler mit der Apparatur verschmelzen und trainieren ihn für die kommenden Kriege. Die Medien, einstmals selbst Kriegstechnologien, leisten bereits in Friedenszeiten die Mobilmachung, so Kittlers Argument. Doch wie könnte eine ›Konversion‹ dieser perfiden Kriegstechnologie namens ›Computerspiele‹ aussehen?

Die Antwort wurde bereits gegeben. Es wäre die Programmierbarkeit der Maschine zu profilieren, ein werkzeughafter ›Missbrauch‹ der Computerspiele. Die Aufmerksamkeit wäre auf die Spiele-Editoren zu lenken, um die Wahrnehmung dafür zu schärfen, dass sich das Potenzial der Computerspiele beileibe nicht auf eine irgendwie geartete »Mobilmachung« engführen lässt. So gibt es keine verlässliche empirische Basis für die These, dass die Spielerinnen gewalttätiger Spiele selbst zu Gewalttäterinnen werden oder mit Vorliebe Dienst an der Waffe verrichten. Die vielen Modifikationen von Computerspielen jedoch, die Menge an zur Verfügung stehenden Foren und Tutorials, die sich mit dem Thema beschäftigen, aber auch die gängige Praxis der *machinimas* jedenfalls sprechen demgegenüber eine eindeutigere Sprache. Es gilt, die Computerspiele als ›Kreativtechnologie‹ zu entdecken.

Es ist sicherlich kein Zufall, dass die Rede vom ›Do It Yourself‹ wieder Konjunktur hat. Unter dem Stichwort vom ›Web 2.0‹ versucht Tim O'Reilly dem Phänomen Rechenschaft zu tragen, dass mit solchen kollaborativen Plattformen wie Flickr, YouTube oder MySpace das Web vom »Publishing« zur »Participation« umschalte (vgl. O'Reilly 2005). Er stellt dies in den größeren Zusammenhang

der Open-Source-Software bzw. der freien Software, die diesen Ansatz bereits seit längerer Zeit verfolge. Was O'Reilly hingegen nicht berücksichtigt, ist, dass der Beginn dieser Entwicklung wesentlich früher anzusetzen wäre. Bereits gegen Ende der 1960er-Jahre konzeptualisierte einer der Architekten des Internets, J.C.R. Licklider, den Computer als ein »Communication Device«. Hierbei gab er der Hoffnung Ausdruck, dass mit dem vernetzten Computer ein Medium entstehe, mit dessen Hilfe sich die gesamte Weltbevölkerung an einem »infinite crescendo of on-line interactive debugging« (Licklider 1990, 40) beteiligen könne. Die Weltgesellschaft, beteiligt an einem permanenten Prozess der Problemlösung mittels vernetzter Computer: Dies mag sicherlich eine naive Vorstellung sein. Jedoch entwickelte sich tatsächlich rund um den Computer eine Kultur der Partizipation, die sich nicht mehr auf den einfachen Nenner vom Produzenten und vom Konsumenten bringen lässt. Dieser Impetus eines ›Do It Yourself‹ wurde von der Debatte über den Computer als Medium in der deutschen Medientheorie der 1990er-Jahren eher verstellt. Die Veränderung in der Medienlandschaft, die der Computer als Medium mit sich bringt, wäre im Sinne dieses partizipativen Momentes nicht bloß in solchen Schlagworten wie Interaktivität, Virtualität und Digitalität zu suchen. Nicht die Überbietung des Menschen durch die Computertechnologie ist das Spannende, daran hat man sich im Zuge der zunehmenden Technisierung der Umwelt längst gewöhnt, sondern vielmehr das, was die gleichen Subjekte mit dieser Technologie trotz allem anstellen.

Anmerkungen

01► http://sauerbraten.org

02► Bei diesem kostenlos zum Download angebotenen Shooter auf Basis von Counter-Strike handelt es sich um den unverhüllten Versuch, Nachwuchs für die US-amerikanischen Streitkräfte zu rekrutieren. Neue Spieler müssen ein aufwendiges Trainingsprogramm absolvieren, bevor sie dann an verschiedenen Missionen teilnehmen können. Ziel ist es, den Spielern an möglichst realistisch nachgebildeten Originalschauplätzen einen Einblick in das »ruhmreiche« Leben eines Elitesoldaten zu ermöglichen (vgl. America's Army Game Leadership Team 2007). Dass America's Army trotz dieses Hintergrundes offenbar auch der Unterhaltung dient, belegt die von offizieller Seite propagierte Anzahl von neun Millionen Accounts (Stand Mai 2007, vgl. ebd).

03► http://www.extremetuxracer.com/

04► Vereinfacht formuliert stellen diese Technologien eine Programmierschnittstelle zur Entwicklung multimedialer Anwendungen bereit, die einen standardisierten Zugriff auf

die entsprechende Hardware wie Grafik- und Soundkarte ermöglichen.

05► Eine recht umfangreiche Auflistung von unter Linux lauffähigen Spielen findet sich auf http://www.holarse-linuxgaming.de.

06► http://www.doomworld.com/classicdoom/ports/index.php

07► http://digita.mame.net/

08► http://red.planetarena.org/

09► http://www.tremulous.net/

10► http://www.alientrap.org/nexuiz/

11► http://www.warsow.net/

12► http://www.crystalspace3d.org

13► http://www.cubeengine.com/

14► Vgl. DGL Wiki: Octree, http://wiki.delphigl.com/index.php/Octree

15► Vgl. http://www.quadropolis.us

16► Vgl. http://www.iddevnet.com/quake4/

17► Die ›Bash‹ ist ein beliebter Kommandozeileninterpreter auf unixartigen Systemen.

18► Im Deutschen hat sich für den Bereich des Films für das englische *script* – oder häufiger: *screenplay* – hauptsächlich der Begriff ›Drehbuch‹ durchgesetzt.

19► Einen Eindruck allein über deutschsprachige Fan Fiction zu The X-Files vermittelt folgende Linksammlung: http://xf.dffi.de/?action=archive.

20► http://eisenstern.com

Bibliografie

America's Army Game Leadership Team (2007): Letter From Leadership. http://www.americasarmy.com/intel/makingof.php (letzter Aufruf am 19.02.2008).

Bekel, M. (2005): Some general mapping tips. http://www.quadropolis.us/node/30 (letzter Aufruf am 19.02.2008).

Berners-Lee, T. (1999): Der Web-Report. Der Schöpfer des World Wide Webs über das grenzenlose Potential des Internets. München: Econ.

Coy, W. (1995): Automat – Werkzeug – Medium. In: Informatik Spektrum, 18, S. 31–38.

Flusser, V. (2002): Die Schrift. Hat Schreiben Zukunft? (5., durchges. Aufl.). Göttingen: European Photography.

Halbach, W. R. (1994): Reality Engines. In: N. Bolz, F. Kittler & G. C. Tholen (Hg.), Computer als Medium. München: Fink, S. 231–244.

Heintz, B. (1993): Die Herrschaft der Regel. Zur Grundlagengeschichte des Computers. Frankfurt am Main/New York: Campus Verlag.

Kittler, F. (1991): Rockmusik – ein Mißbrauch von Heeresgerät. In: T. Elm & H. H. Hiebel (Hg.), Medien und Maschinen. Literatur im technischen Zeitalter. Freiburg: Rombach, S. 245–257.

Kittler, F. (1992): Gleichschaltungen. Über Normen und Standards der elektronischen Kommunikation. In: K. P. Dencker (Hg.), Interface 1. Elektronische Medien und künstlerische Kreativität. Hamburg: Verlag Hans-Bredow-Institut für Rundfunk und Fernsehen, S. 175–183.
Licklider, J.C.R. (1990): The Computer as a Communication Device. In: R. W. Taylor (Hg.), In Memoriam: J.C.R. Licklider 1915–1990. http://memex.org/licklider.pdf, S. 21–40
McLuhan, M. (1995): Die Gutenberg-Galaxis. Das Ende des Buchzeitalters. Bonn/Paris/Reading (Mass.): Addison-Wesley.
O'Reilly, T. (2005): What is Web 2.0? Design Patterns and Business Models for the Next Generation of Software. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html (letzter Aufruf am 19.02.2008).
Poole, S. (2000): Trigger Happy. The Inner Life of Videogames. London: Fourth Estate.
Schelhowe, H. (1997): Das Medium aus der Maschine. Zur Metamorphose des Computers. Frankfurt am Main/New York: Campus Verlag.
Tholen, G. C. (2002): Die Zäsur der Medien. Kulturphilosophische Konturen. Frankfurt am Main: Suhrkamp.
Vatton, I. (2006): Welcome to Amaya. http://www.w3.org/Amaya/ (letzter Aufruf am 19.02.2008).
Winkler, H. (1994): Flogging a dead horse? Zum Begriff der Ideologie in der Apparatusdebatte, bei Bolz und bei Kittler. http://wwwcs.uni-paderborn.de/~winkler//flogging.html (letzter Aufruf am 19.02.2008).

Gameografie

Alien Arena, http://red.planetarena.org/
America's Army (MOVES Institute/United States Army 2002)
Counter-Strike (Minh Le & Jess Cliffe/EA Games 2001)
Crystalcore, http://www.crystalspace3d.org
Cube, http://www.cubeengine.com/
Doom (id Software/id Software 1993)
Eisenstern, http://eisenstern.com
Nexuiz, http://www.alientrap.org/nexuiz/
Extreme Tux Racer, http://www.extremetuxracer.com/
Quake (id Software/Activision 1996)
Quake 4 (Raven Software/id Software 2005)
Sauerbraten, http://sauerbraten.org
Tremulous, http://www.tremulous.net/
War§ow http://www.warsow.net/
Wolfenstein 3D (id Software/Apogee Games 1992)