Analyse und Erprobung der Möglichkeiten zukünftiger Webstandards zur Gestaltung von Browserspielen

Schriftliche Prüfungsarbeit für die Bachelor-Prüfung des Studiengangs
Angewandte Informatik an der Ruhr-Universität Bochum

durchgeführt am
Lehrstuhl für Wirtschaftsinformatik
Ruhr-Universität Bochum

vorgelegt von

Andrae, Mark

Abgabedatum
22.03.2011

1.Prüfer: Prof. Dr. Roland Gabriel
2.Prüfer: Dr. Peter Weber

Inhaltsverzeichnis

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
BVDW	Bundesverband Digitale Wirtschaft e.V.
BIU	Bundesverband Interaktive Unterhaltungssoftware
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IGDA	Internation Game Developers Association
PC	Personal Computer
SVG	Scalable Vector Graphics
URL	Uniform Resource Locator
WaSP	Web Standards Project
WEBGL	Web Graphics Library
WHATWG	Web Hyptertext Applications Technology Working Group
W3C	World Wide Web Consortium
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language

Einleitung

1.1 Motivation

Das Spielen begleitet den Menschen das ganze Leben lang. Bei Kindern findet viel der motorischen und kognitiven Entwicklung mit Hilfe des Spielens statt, im Erwachsenenalter bieten Gesellschaftsspiele oft den Rahmen für soziale Interaktion und im hohen Alter werden Spiele oft zusätzlich zum Erhalt von geistigen Fähigkeiten verwendet. Dabei sind Spiele keine Erscheinung der Neuzeit. Spiele werden im Allgemeinen als fester Teil der kulturellen Entwicklung des Menschen gesehen. Archäologische Funde wie z.B. die des „königlichen Spiels von Ur“, ein altertümliches Brettspiel, aus dem 26. Jahrhundert vor Christus, unterstützen diese Annahme[1].

Dass neue Technologien in der kulturellen Entwicklung der Menschheit auch spielerisch erschlossen werden, ist somit wenig verwunderlich. So sind die Anfänge der Computerspiele auf erste Versuche mit Supercomputern an Universitäten in den späten 50er und frühen 60er Jahren des zwanzigsten Jahrhunderts zurückzuführen[2]. Aus diesen ersten Versuchen ist inzwischen eine Unterhaltungsindustrie entstanden, die pro Jahr allein in Deutschland mit Video- und Computerspielen mehr als 1,5 Milliarden Euro umsetzt und sich weiter im Wachstumstrend befindet. [3]

„*Computerspiele haben mehr für die Verbreitung von Computern getan, als jede andere Anwendung*“[4], wird Nolan Bushnell, Gründer des Unterhaltungselektronikunternehmen Atari, gern zitiert, um die Rolle der Computerspiele bei der Integration der Computer in die Alltäglichkeit der modernen Gesellschaft auszudrücken. Dabei sind es nicht nur Computer, die immer selbstverständlicher werden, sondern auch darauf aufbauende Technologien, wie zum Beispiel das Internet.

Es verwundert nicht, dass auch beim Internet mit der vorhandenen Technologie gespielt wird. So gibt es z.B. eine Form von Computerspielen, die sich ausschließlich auf die Technologien des Internets beschränkt und den Internetbrowser als alleinige Schnittstelle benutzt. Diese Spiele sind die sogenannten Browserspiele.

---

[1] Vgl. British Museum - The Royal Game of Ur

[2] Vgl. Magdans (2008, S.11f)

[3] Vgl. Bundesverband Interaktive Unterhaltunssofware e.V.(2011)

[4] Baumgärtel (1998)

Die Entwicklung des Internets selbst drückt sich in der Weiterentwicklung der zugrunde liegenden Standards und Spezifikationen aus. Seit Oktober 2009 hat das World Wide Web Consortium (W3C) beschlossen bestehende Arbeitsgruppen und Ambitionen zur Schaffung einer neuen Stufe des Standards HTML zusammenzulegen. Diese 5. Version des Standards ist im Allgemeinen unter dem Begriff HTML5 bekannt. Das neue an diesem Standard ist die native Unterstützung von sogenannten „Rich Media"-Fähigkeiten, mit denen vor allem die Unterstützung von multimedialen Inhalten wie Video, Audio und Animationen gemeint sind.

1.2 Ziel

Das Ziel dieser Arbeit ist, die zusätzlichen Möglichkeiten der neuen Spezifikationen, der dem Internet zugrunde liegenden Auszeichnungs-, Formatierung und Programmiersprachen vorzustellen und auf die Verwendbarkeit bei der Erstellung von Browserspielen zu prüfen. Hierbei werden ökonomische Aspekte der Nutzung ebenfalls beleuchtet und abgewägt. Von besonderer Bedeutung ist hierbei, dass es sich bei den verwendeten Technologien um offene Standards handelt und nicht um proprietäre Lösungen, also Lösungen, die sich leicht von Dritten implementieren lassen und lizenzrechtlich keinen Auflagen unterliegen.

1.3 Vorgehensweise

Um dieses Ziel zu erreichen wird zunächst ein Überblick über die bestehenden Begrifflichkeiten, Technologien und Marktgegebenheiten gegeben. Die Begrifflichkeiten umfassen sowohl bestehende Definitionen des Begriffes „Browserspiel", als auch die Abgrenzung zu anderen Computerspielformen.

Daraufhin werden die neuen Fähigkeiten des entstehenden Standards näher erläutert. Hierbei werden ebenfalls die grundsätzlichen Techniken für Browserspiele basierend auf den neuen Standards erläutert und zwischen zwei technischen Möglichkeiten für die Komposition von Browserspielen abgewägt.

Mit diesen Grundlagen wird prototypisch ein Browserspiel entworfen, das an das Computerspiel-Genre des Adventures angelehnt ist.

Geschlossen wird die Arbeit mit einer Zusammenfassung der Ergebnisse und einem Ausblick auf momentane Entwicklungen und Möglichkeiten weiterer Betrachtungen und Forschung.

2 Browserspiele – Formen, Techniken und Markt

2.1 Allgemein

2.1.1 Einführung

Da Browserspiele eine Spezialform von Computerspielen sind, genauer gesagt eine Differenzierung sogenannter Onlinespiele, ist es zunächst nötig den Begriff Computerspiele und Onlinespiele genauer zu erläutern. Auch auf die Taxonomie vom Computer- und Videospielen wird hier kurz eingegangen.

2.1.2 Begrifflichkeiten und Abgrenzungen

Computerspiele grenzen sich im allgemeinen Sprachgebrauch von Videospielen ab. Während mit Videospielen meist Spielekonsolen mit dazugehörigen Spielen gemeint sind, wird mit dem Begriff Computerspiel ein hauptsächlich auf Personal Computern gespieltes digitales Spiel verbunden. Eine gewisse sprachliche Ungenauigkeit ist hier aus dem Umstand zu erklären, dass der englische Begriff „video game“ eher als Synonym für elektronische Spiele aller Art zu sehen ist und die Sprachkultur, die um Video- und Computerspiele entstanden ist, sehr stark von englischsprachigen Begriffen geprägt ist. Als das erste „echte“ Computerspiel wird das Spiel „Pong“ für den Amiga- Heimcomputer aus dem Jahre 1972 gesehen. Die heutige Güte von Computerspielen wurde aber erst mit PCs erreicht die im Stande waren multimediale Inhalte darzustellen[5]. Unter Computerspielen versteht man also „[…]Spiele, die auf einem handelsüblichen PC spielbar sind.“ [6]

Anders als bei Film oder Literatur werden Computer- und Videospiele nicht thematisch in verschiedene Genres unterteilt, sondern nach vorherrschender Interaktionsmöglichkeiten und Spielmechanik. Als wichtigste Genres werden oft die Genres Simulation, Strategie, Action und Rollenspiel genannt[7]. Es kann noch weiter ausdifferenziert werden und auch zu Mischformen zwischen fast beliebigen Genres kommen. Zusätzlich ist zu bemerken, dass der Stand der Technik und des Designs von Computer- und Videospielen sich schnell verändert, was eine eindeutige Taxonomie zusätzlich erschwert.[8]

Als Onlinespiele versteht man nun Computer- oder Videospiele, die „mit Hilfe einer Datenverbindung zwischen einem oder mehreren Rechnern gespielt

[5] Vgl. Frey (2004, S.18f)

[6] Schultheiss (2009, S.18)

[7] Vgl. Apperley (2006)

[8] Vgl. Crawford (1982)

werden“[9]. Das heißt alle Computer- und Videospiele, unabhängig vom Genre, die mittels Datenübertragung über Internet oder lokale Netze gespielt werden, sind „Onlinegames“. Da diese Beschreibung für eine Fülle von verschiedenen Spielrichtungen gilt, und je nach Betrachtungswinkel anders ausgelegt werden kann, versucht eine Studie, die am Hans-Bredow-Institut durchgeführt wurde, hier genauer zu präzisieren. Hier wird vor allem zwischen Browserspielen, „Massively Multiplayer Online Games“, „LAN-Spielen“, „Pervasive Games“, „Passive Multiplayer Online Games“, „E-Sport“ und „Virtuelle Welten“ unterschieden[10]. Browserspiele, in dieser Studie „Browser-Games“ genannt, sind laut Studie „[…] Onlinespiele, die ohne Download bzw. Installation eines Datenträgers auskommen, sondern den Web-Browser als Schnittstelle zwischen Spieler und Spielwelt nutzen“[11]. Der Browserspiele Entwickler und Publisher Bigpoint definiert ähnlich: „Der Begriff „Browsergames“ bezeichnet Online-Spiele, die rein online, also ohne vorherigen Download oder Installation gespielt werden können. […] So ist jedes Spiel von jedem beliebigen Ort aus ganz einfach via URL aufrufbar […]“[12]. – So kommt es vor allem auf ein Spielvergnügen ohne vorherige Installation des Spiels in Verbindung mit einem Webbrowser als Benutzerschnittstelle an.

Eine präzisere Definition lässt sich in einem White Paper der Internation Game Developers Association, für das Synonym „Web game“ finden:

“Web game: A game launched via a web page with no prior installation of software required. This category does not include games that are downloaded to the user’s hard-drive and run outside of the web-browser, but it does include games launched from a web page that might require/installation of a general or custom ActiveX control. Common examples of this are the Flash™, Shockwave™ and Java™ games found on thousands of websites, as well as C++ games delivered via a custom ActiveX control.” [13]

Somit spielt es keine Rolle, ob vor dem Spielen eines Browserspiels ein Browser-Plug-In zur Erweiterung der Funktionalität des Browsers vorgenommen werden muss, solange das Spiel innerhalb des Browsers gespielt wird.

---

9 Vgl. Schmidt, Dreyer, & Lampert (2008, S.7)

10 Vgl. Schmidt, Dreyer, & Lampert (2008, S.7)

11 Vgl. Schmidt, Dreyer, & Lampert (2008, S.13)

12 Bigpoint GmbH (2011)

13 (The IGDA Casual Games SIG (2006)

Als eine neue Form der Distribution von Computer- und Videospielen wird im Zusammenhang mit Browserspielen oft das sogenannte „Cloud gaming“ oder auch „Games-On-Demand“-Konzept genannt. Bei diesem Konzept wird der „Cloud Computing“– Ansatz, bei dem IT-Infrastrukturen dynamisch über das Netzwerk bereitgestellt werden, auf installationsgebundene Computer- und Videospiele übertragen. Somit werden die Spiele auf speziellen Servern installiert und ausgeführt. Die vom Spiel generierten Video- und Audiodaten werden zum Spieler übertagen und von dessen Rechner nur die Eingabedaten zur Spielsteuerung an den Server übermittelt.[14] Die Verbindung zum Browserspiel entsteht dadurch, dass der Abruf eines solchen Spiels durch die multimediale Funktionalität des Browser geschehen kann und somit das Spiel scheinbar im Browser gespielt wird. Die eigentlichen Spiele werden jedoch nicht als Browserspiele entworfen, die Technik setzt eine Installation des Spiels auf einem Server voraus und die Spielmechanik zieht keinen Nutzen aus der Browserschnittstelle. Der Browser wird vielmehr nur als Übertragungsmedium benutzt. Somit wäre der Zugang zum Spiel auch durch eine weitere Clientapplikation möglich, die den Browser ersetzt. Daher kann beim „cloud gaming“ nicht von Browserspielen gesprochen werden. Vielmehr ist hier eine alternative Distributionsstrategie zu erkennen, die einem Spieler den Installationsaufwand abnimmt und selbst nicht als eigenständige Spielform verstanden wird.

2.1.3 Formen von Browserspielen

In der Kategorie der Browserspiele gibt es zwei Hauptrichtungen, die sich durch die Art der Nutzung unterscheiden. Neben, auf längerfristiges Spielen angelegte Spiele, deren Spielgeschehen persistent auf einem Server stattfindet, gibt es die auf kurzfristigen Spielspaß ausgelegten Spiele, deren Spielgeschehen hauptsächlich auf den Rechner des Spielenden berechnet wird. Diese kurzfristigen Spiele werden auch oft „Casual Browsergames“ genannt.

Letztere sind oft Umsetzungen von Spielautomatenklassikern und setzten vermehrt auf Plug-Ins wie Adobe Flash und Java-Applets, um Grafiken, Animationen und Geräuscheffekte umzusetzen. Charakteristisch für diese Spiele sind relativ einfache Grafik und Steuerung und eine nicht persistente Spielwelt.[15] Oft findet man Spiele dieser Art auf spezialisierten Seiten, die die Sammlung, Kategorisierung und Bereitstellung der Spiele mit einem System zur Benutzerbewertung verbinden.

---

[14] Vgl. Onlive.com

[15] Vgl. Schultheiss (2009), S.30

Langfristige Browserspiele haben oft ein strategisches Motiv. Hauptthema eines solchen Spiels ist meist der Aufbau einer Organisation oder eines Herrschaftsraums und der Wettstreit um virtuelle Gebiete und Ressourcen. Technisch wird hier vor allem auf den aktuellen HTML-Standard und serverseitig auf Skriptsprachen und Datenbanken zurückgegriffen. Neuere Spiele setzten vermehrt auf AJAX (**A**synchronous **J**avaScript **a**nd **X**ML) um asynchron Daten zu übertragen und ein flüssigeres Spielerlebnis zu bieten. Die Verwendung von Plug-Ins für multimediale Inhalte wird auch immer beliebter.

Das langfristige Browserspiel „SOL“ wird allgemein als ältestes Browserspiel dieser Art angesehen und startete schon im Jahre 1995. Bei „SOL“ handelt es sich ein persistentes Strategiespiel, bei dem der Spieler einen fremden Planeten besiedelt und eine Kolonie aufbaut. Hierfür hat der Spieler pro Tag eine gewisse Anzahl an Zügen in denen er Ressourcen und Geld erwirtschaften und neue Anlagen bauen kann. Anders als bei neueren Spielen dieser Art, die eine Spielrunde meist nach mehreren Monaten beendet, ist die Spielwelt dauerhaft stabil. Eine Runde endet hier für einen Spieler erst beim Erreichen einer gewissen Ausbaustufe.

Neben diesen kurz- und langfristigen Browserspielen nennt das englischsprachige Wikipedia virtuelle Haustierspiele, die weniger als Spiele und mehr als Simulationen gesehen werden können, technisch jedoch eher mit langfristigen Browserspielen zu vergleichen sind.

Eine Integration von Elementen aus sozialen Netzwerken macht aus Browserspielen oft „Social Games“, obwohl dieser Begriff für seinen fehlenden sozialen Bezug in der Kritik steht.[16]

Nutzung (Spielwelt)		
Architektur	**Langzeit** (persistente Spielwelt)	**Kurzzeit** (nichtpersistente Spielwelt)
Client (herunterladbar)	**Langzeit-Clientgame** (z.B. Silkroad Online)	**Casual-Clientgame** (z.B. Bejeweled)
Browserbasiert (z.B. IE, Firefox, Safari)	**Langzeit-Browsergame** (z.B. Travian)	**Casual-Browsergame** (z.B. Slingo Millenium)

Abbildung 1: Kategorien von internetbasierten digitalen Spielen [17]

[16] Vgl. Blow (2011)

[17] Justus & Schultheiss (2010)

2.2 Techniken

In diesem Abschnitt wird auf die verschiedenen Techniken eingegangen, die heutzutage für die Erstellung von Browserspielen verwendet werden.

2.2.1 HTML, CSS und JavaScript

Die Hypertext Markup Language (HTML), ist eine Auszeichnungssprache zur Strukturierung von Inhalten, Hyperlinks, Bildern und weiteren Medienobjekten in Dokumenten. Sie ist die zugrunde liegende Sprache des World Wide Webs und wird von Webbrowsern interpretiert und angezeigt. Webbrowser, Computerprogramme zur Anzeige von Dokumenten und Daten, werden von verschiedenen Herstellern entwickelt. Somit ist ein gewisser Grad an Standardisierung nötig. Dieser Aufgabe und der Weiterentwicklung der Auszeichnungssprache hat sich das World Wide Web Consortium, kurz W3C, angenommen. Gründer und Vorsitzender des W3C ist der Erfinder von HTML selbst, Tim Berners-Lee, welcher auch Mitautor der Spezifikation für das Protokoll zur Übertragung von HTML-Dokumenten dem Hypertext-Transfer-Protokolls, kurz HTTP, ist. Wichtig bei Entwicklungen des W3C ist, dass alle Entwicklungen öffentlich stattfinden und alle entstehenden Patente unentgeltlich von der Öffentlichkeit genutzt werden dürfen.

Begonnen hat die Entwicklung von HTML schon 1992. Diese erste Version wurde bis zum Jahre 1999 zu HTML 4.01 weiterentwickelt. Teil der Entwicklung war die Integration einer deklarativen Stylesheet-Sprache für strukturierte Dokumente, den Cascading Style Sheets (Abk.: CSS). Des Weiteren wurden clientseitige Skriptsprachen, wie z.B. JavaScript und die Möglichkeit der Einbindung von Applets und Objekten von Plug-Ins eingeführt. Nach 1999 verlief die Entwicklung von HTML aus mehreren Gründen schleppend.

CSS wurde parallel zur Entwicklung von HTML vorangetrieben. Ziel von CSS ist es den Inhalt von HTML-Dokumenten von der Darstellung zu trennen, so dass die HTML-Dokumente zugänglicher, flexibler und leichter auf spezielle visuelle Wünsche anzupassen sind. So kann durch CSS beispielsweise Schriftart, Schriftfarbe, Ausrichtung, Rahmen und Abstände von Objekten genau festgelegt und das Überlappen und die relative Ausrichtung von Objekten ermöglicht werden. Aktuelle Spezifikationen arbeiten an der 3. Hauptversion von CSS, CSS 3, in der auch Objektmanipulationen und Animationen untergebracht werden sollen.

JavaScript ist die hauptsächlich eingesetzte Skriptsprache in Webbrowsern um DOM-Scripting zu erlauben. Bei dem Document Object Model (DOM) handelt es sich um eine standardisierte Schnittstelle für den Zugriff auf Objekte innerhalb

von HTML-Dokumenten. Mithilfe von JavaScript kann das HTML-Dokument dynamisch im Browser nach Aufruf verändert werden. Der Sprachkern von JavaScript wird als ECMAScript standardisiert und seit Erscheinen 1995 stetig weiterentwickelt. Die neuste Version zum Zeitpunkt dieser Arbeit ist die Version 1.8.5, die dazu standardisierte ECMASkript-Version trägt die Versionsnummer 5.

DHTML als Kurzform für dynamisches HTML dient als Sammelbegriff für Techniken, die mit Hilfe von HTML, CSS und JavaScript Webseiten dynamisch je nach Benutzereingaben manipulieren. Spiele auf Basis von diesen Techniken fallen somit zweifelsfrei unter diesen Sammelbegriff.[18]

```
1  <html>
2    <head>
3      <title>HTML</title>
4      <style type="text/css">
5       body{
6       color: blue;
7       }
8      </style>
9    </head>
10   <body >
11     <p>Hello World!</p>
12     <script type="text/javascript">
13         alert("Hello World!");
14     </script>
15   </body>
16 </html>
```

Abbildung 2: Aufbau eines typischen HTML-Dokuments

2.2.2 Flash, Java und Weitere

HTML ermöglicht es Objekte einzubinden deren Datentypen nicht notwendigerweise nativ von einem Browser verstanden werden können. Um solche Daten zu verarbeiten, kann ein Browser mithilfe von Plug-Ins erweitert werden. Auf diesem Weg kann ein Browser um Funktionalitäten erweitert werden, die sonst nicht anders umgesetzt werden könnten. Webanwendungen die diese Technologien verwenden werden auch oft „Rich Internet Applications“ genannt. Die meisten Technologien sind hierbei unabhängig von dem Betriebssystem auf dem der Browser ausgeführt wird.

Die bekannteste, und mit zwischen 85- und 95-prozentiger Marktdurchdringung[19] am meisten verbreitete Lösung, ist die von der kalifornischen Firma Adobe Systems vertriebene Adobe Flash Technologie. Diese Technologie wird vor allem für die Darstellung von multimedialen und interaktiven Inhalten benutzt.

[18] Vgl. SelfHTML.org

[19] Vgl. Adobe.com und Riastats.com

Entwickelt wird Flash schon seit 1996, damals von der amerikanischen Firma Macromedia, die 2005 durch Adobe aufgekauft wurde. Durch die lange Entwicklungszeit ist die Entwicklungsumgebung „Adobe Flash Professional“ weit fortgeschritten und Dank der Bündelung mit dem Bildbearbeitungsprogramm Photoshop, ist Sie vor allem bei Designern und Künstlern beliebt. Durch die produkteigene Programmiersprache ActionScript, die wie JavaScript den ECMAScript-Sprachkern nutzt, wurde Flash auch schnell bei der Entwicklung von Browserspielen beliebt. Casual-Browsergames die die Flash Technologie verwenden, werden auch „Flashgames“ genannt. Kritik am Flashplayer, dem Browserplugin zur Darstellung von Flash-Inhalten, konzentriert sich vor allem auf Sicherheitsprobleme, Stabilitätsprobleme und schlechte Performance.

Abbildung 3: Bejeweled, eines der erfolgreichsten Flashgames

Der Erfolg der Programmiersprache Java wird oft den Java-Applets zugeschrieben. Diese ermöglichen es fast beliebige Java-Programme in eine Webseite einzubinden. Java selbst ist auf keine speziellen Anforderungen spezialisiert und somit eine „general purpose“ -Programmiersprache. Java-Applets werden oft im universitären Bereich eingesetzt, es sind aber auch diverse Spiele auf Basis von Java–Applets zu finden. So basiert zum Beispiel das Langzeit-Multiplayer-Browserspiel Runescape auf Java-Applets.[20]

Der Betriebssystemhersteller Microsoft veröffentlichte mit Silverlight im April 2007 eine eigene Lösung zur Erstellung von Rich Internet Applications. Diese

[20] Vgl. Runescape.com

Lösung basiert hierbei auf einer abgespeckten Version der .NET-Plattform, einer Laufzeitumgebung vergleichbar mit der Java-Laufzeitumgebung.

Neben diesen gibt es mehrere Plug-Ins von Drittherstellern, die die Nische, der fehlenden oder überkomplizierten 3D-Grafikunterstützung füllen wollen. Bekannte Plug-Ins aus diesem Bereich sind der Unity Web Player, Panda 3D Web-Runtime oder Tourque 3D Web-PlugIn. Hierbei scheint der Unity Web Player zurzeit der bedeutendste zu sein, da die dazugehörige Entwicklungsumgebung Unity3D laut Aussage der Entwicklungsfirma bei ca. 250.000 Spieleentwicklern eingesetzt wird und das Plug-In bei über 35 Millionen Benutzern installiert sein soll.[21]

2.2.3 Serverseitige Skriptsprachen und Datenbanken

Um langfristige Browserspiele zu persistieren, setzten die meisten Spiele auf eine Client-Server-Architektur. Das Spielgeschehen findet auf dem Server statt und die Spieler haben mit ihren Browsern als Clients die Möglichkeit verschiedene Aktionen anzustoßen, die auf dem Server ausgeführt werden. Diese Art der Architektur entspricht der der meisten Webapplikationen und wird somit technisch auch mit aktuellen Techniken aus diesem Bereich umgesetzt. Somit meist der Fall, dass auf einem Webserver liegende Programmskripte bei einer Aktion aufgerufen und interpretiert werden. Innerhalb eines Aufrufs wird dann meist eine Datenbankverbindung aufgebaut, die spielrelevanten Daten verändert und daraufhin die Verbindung zu Datenbank geschlossen. Datenbanken werden hier vor allem als Lösung für konkurrierende Zugriffe, sichere Transaktionen und strukturierte Datenhaltung verwendet. Beliebte Skriptsprachen in diesem Bereich sind PHP, Perl, Ruby oder Python. Es gibt auch Lösungen auf Basis von Programmiersprachen die nicht zu den Skriptsprachen gehören. Meist wird dann jedoch eine spezialisierte Serverumgebung benötigt. Als Datenbanken werden oft die relationale Datenbanksysteme Mysql oder Sqlite verwendet. Je nach verwendeter Programmiersprache werden aber auch Lösungen, die eher der verwendeten Plattform entsprechen benutzt. So ist zum Beispiel im Fall von C# - Lösungen mit der .Net-Plattform der Microsoft SQL-Server eine sinnvolle Wahl.

[21] Vgl. Unity3d.com

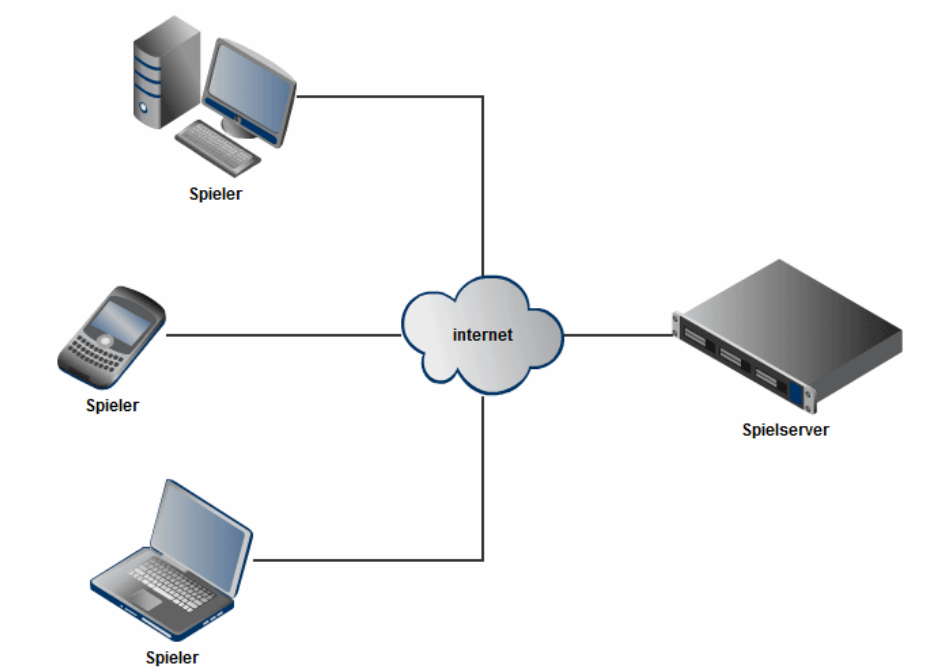

Abbildung 4: Veranschaulichung der Client-Server-Architektur bei Langzeitbrowserspielen

2.3 Markt

2.3.1 Markt

Charakteristisch für die Browserspielbranche sind relativ geringe Entwicklungskosten, kurze Entwicklungszeit und ein langer Produktlebenszyklus. Das Browserspiel „SOL“ ist z. B. bereits 18 Jahre auf dem Markt und wird noch immer täglich von ca. 6500 Spielern gespielt.[22]

Laut einer Umfrage des Bundesverbandes Interaktive Unterhaltungssoftware e.V. aus dem Februar 2010 spielen in Deutschland etwa 12,4 Millionen Personen ab 10 Jahren Online- und/oder Browserspiele. Nach dieser Studie zahlen insgesamt 2,4 Millionen Personen durchschnittlich 11,27 € für die Teilnahme an Online- bzw. Browserspielen. Das entspricht einem Prozentsatz von 19,35% der Onlinespieler. Für Browserspiele wird eine Zahl von insgesamt 9,9 Millionen Spielern angegeben. Eine Zahl von 500.000 Personen wird angegeben, die für Browserspiele eine Gebühr entrichten. Mit einer durchschnittlichen Ausgabe von 7,76 € pro Monat wird somit ein angebliches Marktvolumen von 46,4 Millionen Euro für Browserspiele in Deutschland errechnet.[23]

Diesen Zahlen wird von Vertretern der Browserspieleherstellern Bigpoint und Gameforge in einem Interview der Zeitschrift „Making Games Magazine“

[22] Vgl. Bjørstad

[23] BIU e.V. (2010)

widersprochen. Sowohl der Gesamtumsatz, als auch die durchschnittlichen Ausgaben pro zahlenden Spieler scheinen sich hier nicht mit den Erkenntnissen der Unternehmen zu decken. Es werden jedoch keine konkreten Zahlen genannt.[24]

Auch die Fachgruppe Connected Games im Bundesverband Digitale Wirtschaft e.V. (BVDW) kommt im eigenen Online Games Report 2010 bezüglich des Umsatzes der Onlinespiele-Branche zu einem anderen Schluss als die BIU-Studie. So sollen laut Studie rund ein Drittel der Onlinespieler für ihr Hobby zahlen und der durchschnittliche zahlende Onlinespieler 28 Euro monatlich für Onlinespiele ausgeben.[25] Durch den dadurch resultierenden fast viermal so großen Gesamtumsatz der Onlinespiele-Branche, kann auch bei den Browserspielen ein deutlich höherer Gesamtumsatz angenommen werden.

Gründe für diese Unterschiede kann die ungenaue Betrachtungsweise innerhalb der BIU-Studie sein, die nur nach monatlichen Gebühren von Browserspielen fragt, jedoch alternative Erlösmodelle außen vor lässt. Umsätze aus der Vergütung für Werbung sind von beiden Studien nicht erfasst.

Weitere Schätzungen des Umsatzes der Online-Game-Branche kann dem Brachenreport der medienboard Berlin-Brandenburg GmbH aus dem Jahr 2008 entnommen werden. Dort wird der Gesamtumsatz bei Online-Games in Deutschland auf ca. 672 Millionen Euro Für das Jahr 2010 geschätzt. Weiter werden die Einnahmen der Spielebranche aus Werbung in und im Umfeld von Computer- und Videospielen allgemein weltweit auf 1 Milliarde Euro geschätzt.[26]

Zunehmende Bedeutung wird vor allem den Social-Games zugeschrieben, deren weltweiter Umsatz 2010 auf eine Millarde Dollar geschätzt wird. [27]

Diese Zahlen sprechen für einen Markt der schon beachtliche Dimensionen erreicht hat, jedoch durch viele verschiedene Gegebenheiten schwer als Ganzes zu erfassen ist.

Mit der Games Convention Online hat diese Branche auch eine eigene Fachmesse, die, als Sie zum zweiten Mal veranstaltet wurde, 630 Fachbesucher anzog.[28]

Bekannte deutsche Entwickler sind Bigpoint, Gameforge, Travian Games, RedMoon Studios und upjers. Bekannteste internationale Firma ist Zynga. Zynga

[24] Vgl. Making Games Magazine (2010)

[25] Bundesverband Digitale Wirtschaft e.V. (2010)

[26] Medienboard Berlin-Brandburg GmbH (2008)

[27] USA Today: "For Social Networks, it's Game On"

[28] Vgl. Gamesconvention.com

ist vor allem durch die Social Games Farmville und Mafia Wars bekannt und macht immer wieder Schlagzeilen durch eine aggressive Marktstrategie und den Aufkauf von vielen kleineren Firmen.[29]

2.3.2 Trends

Trotz teilweise als stagnierend angesehenen Umsätzen der Onlinespiele-Sparte, steigt der Umsatz von Browserspielen in allen Studien.

Auch kann eine zunehmende Professionalisierung der Branche beobachtet werden. So wird beispielsweise immer öfter die aus der normalen Computerspielbranche bekannte Aufteilung in Publisher und Entwicklungsstudio praktiziert.

Zunehmend ist ebenfalls der Trend zu größeren Investitionen. So investierte der Browserspielehersteller Bigpoint mehr als zwei Millionen Euro in eine Onlineadaption der „Grand Theft Auto"-Reihe namens Poisonville. Wegen technischen Schwierigkeiten bei der Umsetzung auf Basis der Java-Applet-Technik, wurde dieses Projekt jedoch eingestellt.[30]

Vor allem der Bereich „Social Games" wird, durch die steigende gesellschaftliche Bedeutung von sozialen Netzwerken, immer mehr zur markttreibenden Kraft der Branche.

Wo Browserspiele nicht an Portale gebunden sind, setzten sich Benutzerverwaltungen mit zusätzlicher Option zur Nutzung von Authentifizierungssystemen wie OpenID, Facebook Connect oder Google Single Sign On durch. Diese ermöglichen es den Spielern sich mit Anmeldedaten aus anderen Systemen anzumelden und ermöglichen es so, dass der Spieler nicht für jedes Spiel neue Anmeldedaten generieren muss.

2.3.3 Erlösmodelle

Für die zunehmende Kommerzialisierung von Browserspielen haben sich mehrere Erlösmodelle entwickelt.

Der wissenschaftliche Mitarbeiter der technischen Universitäts Ilmenau, Daniel Schultheiss unterscheidet in einer Studie zu Langzeit-Browsergames zwischen

[29] Vgl .Zynga.com

[30] Vgl. browsergames.de (2011)

Werbefinanzierung, Ingame-Advertising, Premium-Account-Systemen, Abonnements und Pay-for-Goods.[31]

Bei der klassischen Werbefinanzierung, wie von anderen Medien bekannt, wird neben dem Spiel Werbung eingeblendet. Dies passiert internettypisch mithilfe von Bannern, Buttons oder Overlays. Bei Flash Games hat sich ein Handel entwickelt, bei dem Spieleentwickler Lizenzen zur Verwertung ihrer Spiele versteigern. Dies passiert auf Plattformen wie flashgamelicense.com. Die Käufer der Lizenzen nutzen diese zur Steigerung der Benutzerzahlen und somit zur Steigerung ihrer Werbeumsätze. Durch einen stark umkämpften Markt und die steigende Verbreitung von Add-Blockern, das sind Browser-Plugins, die Werbung ausblenden, sind hier kaum noch große Gewinne zu erwarten.

Eine neuere Idee aus dem Bereich der Werbung ist die Werbung direkt in das Spiel zu integrieren. Somit werden Werbebotschaften als Teil der Spielwelt wahrgenommen. Wichtig ist hier eine realistische Vermittlung innerhalb des Spielkonzeptes. Beispiele sind Bandenwerbung an einem virtuellen Fußballplatz, geschicktes Productplacement oder humorvoll aufgemachte Parallelen zu realen Produkten. Diese Art der Werbung wird ein großes Potenzial zugesprochen und bei Browserspielen immer häufiger eingesetzt.

Premium-Account-Systeme sind eine sehr beliebte Form der der Wertschöpfung von Langzeit-Browserspielen. Hierbei steht dem Spieler die Option offen, durch das Zahlen eines Beitrags, seinen sonst kostenlosen Zugang zum Spiel zu einem Premiumzugang aufzuwerten. So erhält er oft Werbefreiheit und Vorteile im Spiel. Dieses System ist Vergleichbar mit dem Freemium-Konzept, bei welchem Internetfirmen eine Basisfunktionalität kostenlos bereitstellen, für weitere Dienste jedoch eine Gebühr erheben. Bei der Entwicklung von Browserspielen muss dann darauf geachtet werden, die Spielebalance zwischen zahlenden und nichtzahlenden Spielern nicht zu stark zu beeinflussen.

Abonnements können wie bei Client-Onlinespielen ebenfalls eine Einnahmequelle sein. Hierbei gibt es Modelle, den Zugang zum Spiel allein zahlenden Kunden zu gewähren. Normalerweise werden Abonnements zusammen mit Premium-Account-Systemen eingesetzt. Die erweiterten Funktionen werden hier also nur monatsweise gewährt. Charakteristisch ist ein Preisnachlass für längere Bindung des Abonnements, so ist ein einmal bezahltes zwölfmonatiges Abonnement meistens deutlich günstiger als zwölf Abonnement für jeweils einen Monat.

Ein weiteres an das Freemium-Konzept angelehntes Konzept ist das Pay-For-Goods-Modell. Hierbei wird jedoch nicht für zusätzliche Funktionen gezahlt,

[31] Vgl. Schultheiss (2009)

sondern für virtuelle Güter direkt im Spiel. Diese Güter können entweder normale Gegenstände innerhalb des Spiels sein, die ein Spieler auch mit normalem Spielen erlangen könnte oder Güter die zahlenden Kunden vorbehalten sind. Vor allem im Bereich der Social Games wächst die Beliebtheit dieses Erlösmodells. Kritisch muss hier gesehen werden, dass bei Spielen, die einen Wettbewerb zwischen Spielern zulassen, zahlende Spieler durch das einsetzten höherer Beträge einen erheblichen Vorteil vor ihren Mitbewerbern erlangen können.

Ein Konzept, das häufig mit den verschiedenen Erlösmodellen verbunden wird ist das Micropayment. Es beschreibt ein Zahlungsverfahren, das ermöglicht kleine Beträge zwischen wenigen Cent und ungefähr. 5 Euro kostengünstig abzurechnen, ohne das für jede Transaktion, wie zum Beispiel bei Kreditkarten üblich, Gebühren anfallen. Browserspielehersteller arbeiten oft mit Firmen zusammen die solche Bezahlsysteme anbieten. Die Anforderungen an solche Bezahlsysteme unterscheiden sich hierbei von Land zu Land. Während die Überweisung bzw. der Bankeinzug eher ein deutsches Phänomen ist, wird in anderen Ländern üblicherweise eher per SMS, Kreditkarten oder Prepaid-Karten bezahlt.[32]

2.3.4 Wirtschaftliche Betrachtung der Techniken

Laut Europäischer Union haben Offene Standards folgende Eigenschaften: der Standard wird von einer Nicht-Profit-Organisation innerhalb eines offenen Prozesses entwickelt, das Spezifikationsdokument wird entweder kostenlos oder gegen eine geringe Gebühr der Öffentlichkeit zugänglich gemacht, der Standard wird lizenzfrei angeboten und die Wiederbenutzung des Standards unterliegt keinen Restriktionen.[33] Dies ist bei HTML und dessen Entwicklung innerhalb des W3C der Fall.

Browserspiele die auf offene Standards setzten müssen daher keine Lizenzgebühren für die eingesetzte Technik entrichten. Anders sieht das aus, wenn Plug-Ins von Drittherstellern genutzt werden. Diese unterliegen meist den Lizenzbedingungen der Plug-In-Anbieter und können je nach Einsatzzweck nur kostenpflichtig für ein kommerzielles Produkt eingesetzt werden.

Die Arbeit mit offenen Standards hat auch oft den Vorteil, dass keine spezialisierte Software zur Generierung der Inhalte gekauft werden muss. So gibt es viele Quelltexteditoren für HTML, die kostenlos auch für kommerzielle Zwecke eingesetzt werden dürfen. Bei Editoren für Plug-In-Inhalte ist dies meist nicht der Fall.

---

[32] Justus & Schultheiss (2010)

[33] Vgl. European Commission (2004)

Dadurch, dass Browserhersteller für das Einhalten der Standards und eine zukünftige Abwärtskomptabilität verantwortlich sind, kann ein Softwareprojekt auf Basis von HTML, zukunftssicher mit den verfügbaren Techniken kalkulieren. Gewisse Vorsicht ist bei nicht finalen Spezifikationen geboten, da sich Details der Spezifikation ändern können.

Ebenfalls von Vorteil ist, dass Sicherheitslücken in Browsern oft sehr viel schneller geschlossen werden, als Sicherheitslücken von Plug-Ins.

Ein weiterer, nicht zu vernachlässigender Nachteil, von Lösungen die stark von Plug-In-Inhalte abhängig sind, ist die fehlende Möglichkeit Plug-In-Daten von Suchmaschinen durchsuchbar zu machen. Hier müssen Anbieter oft eine doppelte Datenpflege vornehmen.

Des Weiteren müssen beim Einsatz von dem Endbenutzer unbekannten Plug-Ins Vertrauensmarketingmaßnahmen durchgeführt werden. Das ist zwar nicht der Fall bei weit verbreiteten Plug-Ins wie Flash, jedoch kann eine zusätzliche Installation von Software immer zu einer Einstiegshürde führen. Besonders gravierend sind solche Hürden, falls ein Benutzer nicht die Kontrolle über die Software auf dem von ihm genutzten Rechner hat. Dies ist z.B. oft bei Arbeitsplatzrechnern der Fall.

Zusätzlich kann momentan die Entwicklung beobachtet werden, dass mobile Geräte, wie Tablet-PCs oder Smartphones, hoch entwickelte Browser mitbringen. Diese unterstützen oft neuste Webtechnologien, jedoch selten Inhalte von Plug-Ins. Durch Einsatz von HTML ist es daher möglich, Applikationen auch auf diesen mobilen Geräten anzubieten. Durch diese Form werden auch mögliche Endgelder an Gerätehersteller für die Prüfung und Verteilung der Applikation auf speziellen Geräten umgangen.

3 Zukünftige Webstandards zur Gestaltung von Browserspielen

3.1 HTML 5

3.1.1 Entstehung

Ungefähr ab Fertigstellung von Version 4.01 von HTML sollte eine Weiterentwicklung sich, nach dem Willen des W3C, näher an der Entwicklung der Auszeichnungssprache Extensible Markup Language, kurz: XML, orientieren. Resultat war XHTML, die eXtensible HyperText Markup Language, die in großen Teilen zu HTML 4.01 kompatibel ist, jedoch strengere Regeln bei der Verarbeitung vorsieht, wie zum Beispiel das Beachten von Groß- und Kleinschreibung. Vorteil sollte die einfache Erweiterbarkeit der Sprache durch die an XML angelehnten Namensräume sein. Dadurch entstanden auch zwei Arten HTML-Dokumente zu parsen. Eine Art ist ein stark fehlerverzeihender Parser der trotz Fehlern im Dokument versucht das Dokument anzuzeigen. Die andere Art zu Parsen brach den Interpretationsvorgang bei einem Fehler komplett ab und zeigt nur eine Fehlermeldung. Diese zweite Art wurde im Allgemeinen als sehr unpopulär empfunden, da schätzungsweise 99 Prozent aller existierenden Seiten mindestens einen Fehler enthielten und so der Endbenutzer bei den meisten Seiten statt der Seite einen Fehler angezeigt bekommen hätte. Deshalb wurde diese Möglichkeit von den meisten Anwendungsentwicklern ignoriert und der „alte“ Standard verwendet. In den weiterentwickelten Versionen von XHTML, wie Versionen 1.1 oder die nie fertiggestellten Version 2.0, ist die Möglichkeit des vergebenden Parsers ausgeschlossen worden. Dies wird als Hauptgrund für die fehlende Akzeptanz von XHTML angesehen. Diese fehlende Akzeptanz führte sogar dazu, dass die jetzt als neu diskutierte 5. Version von HTML lange Zeit außerhalb des W3C entwickelt wurde. Ungefähr Mitte 2004 wurde die Web Hyptertext Applications Technology Working Group (WHATWG) gegründet, die außerhalb des W3C sich Anforderungen abseits von XML und teilweise entgegengesetzt zu bestehenden Betriebssystem-Schnittstellen annahm. Als im Oktober 2006 dann feststand, dass die Arbeit der WHATWG weit fortgeschrittener war als die Weiterentwicklung von HTML innerhalb des XHTML 2.0 Standards der W3C, wurde von Tim Berners-Lee selbst angekündigt, dass das W3C und die WHATWG zusammen an einer neuen Version von HTML arbeiten würden, dem HTML 5 Standard. Die Arbeiten an diesem Standard sind inzwischen soweit fortgeschritten, dass die meisten Browserhersteller viele der neuen Funktionen bereits umsetzten konnten oder in der nächste Version des betreffenden Browsers umsetzten werden. Im Oktober 2009 wurde dann die

Arbeit an dem XHTML 2.0 Standard eingestellt, ohne dass es dieser zur Marktreife gebracht hat.[34]

3.1.2 Übersicht über aktuellen Stand

Zum Zeitpunkt dieser Arbeit ist die Spezifikation von HTML5 als sogenannter „working draft“, also der Arbeitsentwurf des für die Spezifikation verantwortlichen alleinigen Autors Ian Hickson öffentlich zugänglich. Dieser umfasst den kompletten Funktionsumfang und alle zulässigen Elemente der Beschreibungssprache, sowie Hinweise zur Benutzerinteraktion, Rendern von Elementen und veralteten Funktionen aus vorherigen Versionen.

Nach Aussage der W3C in einer Presseerklärung vom 14. Februar 2011 wird die Arbeit an HTML5 im Mai 2011 in den sogenannte „Last Call“-Zustand erhoben. Dieser ist als Einladung an die Öffentlichkeit zu verstehen, die technische Stichhaltigkeit der Spezifikation zu prüfen. Für die Fertigstellung des HTML5 Standards als offizielle Empfehlung des W3C wird das Jahr 2014 angepeilt.[35]

Am 18. Januar 2011 stellte der W3C auch ein neues Logo vor, welches für alle neuen HTML5-Techniken, sowie mit Zusatzsymbolen ergänzende Techniken wie CSS3, SVG und weitere stehen sollte. Webseiten, die auf die neuen Techniken setzten können dieses Logo lizenzfrei einbauen um auf die besondere Technik aufmerksam zu machen.

Abbildung 5: Das offizielle Logo des HTML5-Standards

Ungeachtet des Status der Spezifikation haben die meisten Browserhersteller begonnen, die neuen Elemente und Funktionalitäten in Ihre Produkte zu implementieren.

So können zum Zeitpunkt dieser Arbeit die neusten Versionen von Internet Explorer, Firefox, Chrome und Webkit, der Open-Source-Grundlage der Safari-Browser, bereits mit vielen neuen Elementen umgehen.

---

[34] Vgl. Pilgrim (2010)

[35] W3C (2011)

3.1.3 Erweiterte und neue Funktionalitäten

Die neuen Funktionen von HTML5 umfassen multimediale Elemente, wie das Audio-, Video- und Canvas-Element. Hinzu kommen Möglichkeiten der Datei- und Datenablage für den Offline-Modus des Browsers, Möglichkeiten Elemente direkt im Browser editierbar zu machen und ein Drag-and-Drop-Schnittstelle. Des Weiteren gibt es Möglichkeiten zum Austausch von Nachrichten zwischen Dokumenten, sowie Möglichkeiten den Browserverlauf zu verwalten und eine erweiterte Datenhaltung im Browser.

Bei den neuen Elementen gibt es auch Elemente, die die Einteilung einer Seite erleichtern sollen, namentlich sich es die Elemente „article“, „footer“, „header“, „section“ und „summary“. Des Weiteren gibt es Elemente zur semantischen Einteilung von Text.

Für Formulare wurden neue Eingabeelemente definiert, die durch diese Definition im Browser validiert werden können und nicht mehr in JavaScript auf das richtige Format überprüft werden müssen. Diese neuen Eingabeelemente sind zum Beispiel Datum und Uhrzeiteingaben, Emailadressen, URLs, Nummern, ein Zahlenbereich, Farben und Telefonnummern.

Funktionen, wie Schnittstellen zum Bereitstellen von Standortinformationen, ein Dateisystem für Webapplikationen und Dateioperationen außerhalb des Browsers werden auch oft im Zusammenhang mit HTML5 genannt, sind aber sind Teile eigenständiger Spezifikationen des W3C. Des Weiteren werden neue Parserregeln eingeführt, die flexibler sind und höhere Komptabilität erlauben.

Neue Attribute für Zeichencodierung und die Möglichkeit den Ausführzeitpunkt von Scripts während des Parsens von Dokumenten bestimmen zu können, vervollständigen das Bild.

In Html5 werden folgende Elemente als nicht mehr zulässig behandelt: „acronym“, „applet“, „basefont“, „big“, „center“, „dir“, „font“, „frame“, „frameset“, „isindex“, „noframes“, „strike“, „tt“, „u“.[36]

3.1.4 Für Spiele relevante multimediale Schnittstellen

Für Spiele relevante multimediale Möglichkeiten bietet das Canvas-Element, das eine Art Zeichenfläche bereitstellt, die mit Hilfe von JavaScript-Befehlen gefüllt werden kann und das Audio-Element mit dessen Hilfe Audio-Dateien geladen und deren Abspielen kontrolliert werden kann. Das Video-Element kann im Spiele-Kontext auch sinnvoll eingesetzt werden. Nicht nur als Möglichkeit Intro-Filme

[36] Vgl. Pilgrim (2010)

oder Ähnliches einzuspielen. Durch die Möglichkeit Videodaten zum Befüllen von Canvas-Elementen zu benutzten können auch Ingame-Werbung oder Hintergründe auf neue Weisen animiert werden.

Das Canvas-Element ist ein Feld, das per JavaScript „bemalt“ werden kann. Der JavaScript-Code greift hierbei direkt auf eine Schnittstelle des Browsers zu, der Grafikoperationen zulässt. Die Grafikoperationen selbst unterscheiden sich je nach angesprochenem Kontext der Schnittstelle. Viele neuere Browser unterstützen bereits die beiden Kontexte „2D“ und „WebGL“. Wobei letzterer eine Möglichkeit zur hardwarebeschleunigten Anzeige von 3D-Grafiken ist.

Der 2D-Kontext bietet Zugriff auf Funktionen für Bildtransformationen wie Skalierung oder Drehungen, Möglichkeiten Bildelemente oder auch nur Ausschnitte aus diesen zu zeichnen, verschiedene Kompositionsarten zu wählen, Lienen zu ziehen, einfache geometrische Formen zu zeichnen und Text zeichnen zu lassen. Hierbei kann ebenfalls die volle Kontrolle über die gewünschten Farben und Strichstärken übernommen werden. Zusätzlich bietet dieser Kontext noch Möglichkeiten der direkten Pixelmanipulation an.

Sehr viel komplizierter in der Anwendung ist der WebGL-Kontext. Er ermöglicht Zugriff auf 3D-Funktionen auf einer sehr niedrigen Abstraktionsebene. So müssen zur Anzeige von 3D-Grafiken mehrerer Pufferspeicher des 3D-Kontext sinnvoll in einem speziellen Datenformat gefüllt werden, Grafikhilfsprogramme geladen und Matrizen berechnet werden. Sicherlich für erfahrene 3D-Programmierer eine einfache Sache, jedoch für die meisten Webprogrammierer eine völlig neue Welt. Glücklicherweise bilden sich relativ viele Open-Source-Projekte, die Bibliotheken für höhere Abstraktionsebenen bereitstellen. Bei ihrer Verwendung muss der Webprogrammierer nur rudimentäre 3D-Kentnisse besitzen.[37]

In beiden Kontexten können bewegte Bilder und damit Animationen geschaffen werden, indem das ausführende JavaScript-Programm jedes einzelne Bild der Animation berechnet und im Millisekundenbereich ausgibt. Hierdurch sind beeindruckende Animationen möglich, jedoch funktioniert davon keine ohne die Ausführung von JavaScript.

[37] Vgl. processingjs oder three.js

Der Audio-Tag ermöglicht es Töne, Musikstücke oder Geräusche als HTML-Elemente einzufügen. Der Video-Tag ermöglicht es Videodateien direkt im Browser abzuspielen. Mit Hilfe des Attributs „controls“ kann der Browser bei beiden Elementen dazu gebracht werden Steuerelemente, wie einen Start/Stop-Button, eine Fortschrittsanzeige und eine Lautstärkeregelung anzuzeigen. Weitere Attribute steuern das automatische Abspielen oder Laden von Audio- und Video-Dateien. Innerhalb des Elements selbst können „source“-Elemente angegeben werden, die als Quellverweise zu den abzuspielenden Audio- und Video-Dateien gesehen werden können. Diese können auch Verweise zu Streaming-Servern enthalten und somit Streaming Media abspielen. Falls mehrere Source-Elemente vorhanden sind, wird das erste in der Reihe abgespielt, das der Browser verarbeiten kann. An diesem Punkt unterscheiden sich viele Browser. Der Browser Firefox unterstützt z.B. ausschließlich lizenzfreie Formate z.B. das Ogg Vorbis-Format, andere Browser erlauben auch Formate wie mp3 oder H.264/MPEG-4, die eine Nutzungsgebühr einfordern. Bis sich hier einheitliche Formate durchsetzen, müssen für Audio- oder Video-Elemente mehrere Datei-Formate vorbereitet werden. Ein vielversprechendes Format in dieser Richtung ist das sogenannte WebM-Format.

Auch die Drag-und-Drop-Schnittstelle kann für Browserspiele interessant werden. Diese Schnittstelle ermöglicht es Browsers Elemente durch anklicken und halten, ähnlich wie von Desktop-Systemen wie Windows bekannt, zu ziehen und auf anderen Elementen abzulegen. Für Spiele erschließen sich so neue Interaktionsmöglichkeiten jenseits von Tastatureingaben und einfachen Klicks. Obwohl Drag-and-Drop auch mit etwas JavaScript-Programmierung selbst zu implementieren ist, ist vor allem die Möglichkeit auch Objekte von außerhalb des Browsers in den Browserkontext zu ziehen, neu.

Weitere Hilfreiche Elemente für Spiele sind die Möglichkeit seine Applikationsseiten im Offlinemodus verfügbar zu machen, JavaScript mit Hilfe von sogenannten Webworker-Threads im Hintergrund auszuführen und die erweiterten Datenhaltungsmöglichkeiten nutzen zu können.[38]

[38] Vgl. Pilgrim (2010)

3.2 CSS3

3.2.1 Entstehung

Vorschläge für Strukturierungssprachen wie CSS gab es relativ kurz nach der Veröffentlichung der ersten HTML-Standards. Als „Väter“ gelten Håkon Wium Lie und Bert Bos. Zwar gab es zur gleichen Zeit weitere Formatierungssprachen mit ähnlichen Zielen, jedoch waren es die Ideen die Präsentation des Dokumentes von mehreren Quelldateien abhängig machen zu können und Formatierungen zwischen mehreren Quelldateien vererben zu können, die CSS zum Durchbruch verhalfen.

Im Dezember 1996 wurde dann die erste offizielle CSS Version, CSS Level 1, als Implementierungsvorschlag vom W3C veröffentlicht. Diesem Vorschlag folgten die meisten Browserhersteller bis heute nahezu vollständig.

Die 2. Version, CSS Level 2, wurde im Mai 1998 veröffentlicht. Neuerungen waren die z.B. pixelgenaue Positionierung von Elementen und die Möglichkeit Elemente in Schichten übereinander zu legen. Bei der Implementierung wurde lange kein Konsens zwischen den verbreiteten Browsern erreicht. Die dadurch entstehenden Schwierigkeiten bei der Umsetzung in der Gestaltung von Webseiten, führten unter anderem auch zu sogenannten CSS-Hacks, die trickreiche Umgehungen, etwa durch Ausnutzen von Fehlern, für solche Gestaltungsprobleme waren. 2002 wurde der Standard überarbeitet. Es wurden Unstimmigkeiten entfernt, aber auch Techniken, die von Browserherstellern nicht umgesetzt wurden, auf spätere Versionen verschoben oder gestrichen. Die neue Bezeichnung war nun CSS2.1. Da dieser Standard immer noch nicht ausreichend umgesetzt wurde, entwickelte das Web Standards Project (WaSP), an Anlehnung an den CSS-Test „Acid“ aus dem Jahre 1998, den CSS2.1-Test „Acid2“. Dieser Test besteht aus einem einzelnen HTML-Dokument, das mit Browsern mit standardkonformer Implementierung ein bestimmtes Bild, in diesem Fall einen Smiley, anzeigen sollte. Mit nicht-standardkonformen Browsern wurde das entsprechende Bild gar nicht oder nur mit Fehlern erreicht. Bei diesem Test scheiterten fast alle der sich damals auf dem Markt befindlichen Browser. Heutzutage implementieren die meisten modernen Webbrowser den CSS2.1 Standard korrekt und liefern beim „Acid2“-Test ein korrektes Bild aus.

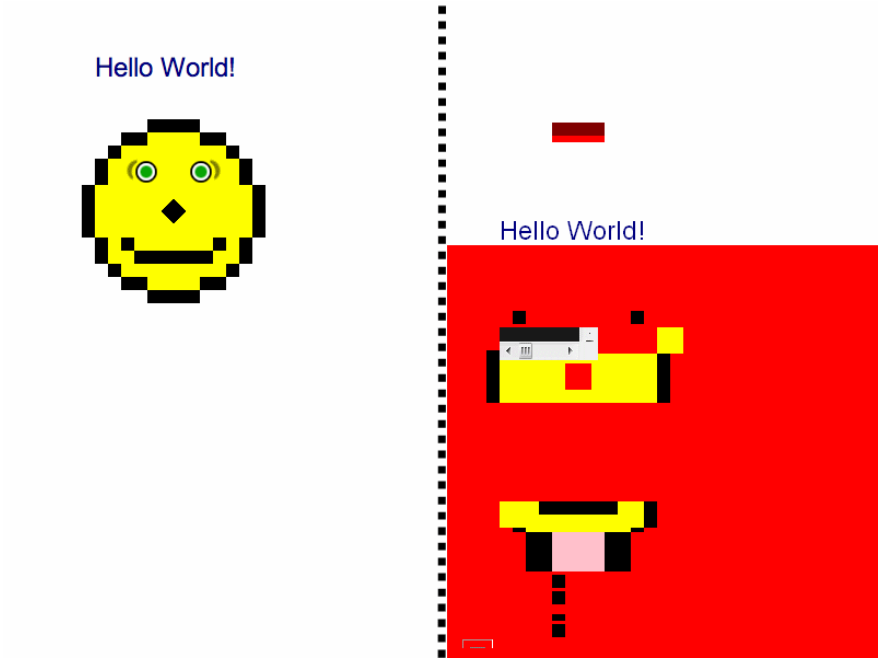

Abbildung 6: Korrektes Ergebnis des Acid2 Test (links) im Vergleich zu dem falschen Ergebnis des Internet Explorer 7.0 (rechts)

Im Jahr 2000 wurde mit der Entwicklung von CSS Level 3 begonnen. Anders als bei den Vorgängern wurde hier ein modulares Vorgehen gewählt, bei dem Teiltechniken nach eigenem Rhythmus und in eigenen Versionsschritten entwickelt werden können. So werden z.B. Selektoren, Farbschemata oder Positionierung in einzelnen Modulen entwickelt.

3.2.2 Aktueller Stand

Nachdem fast alle aktuellen Browser den CSS2.1 Standard unterstützen, wird allmählich mit der Implementierung von CSS3 begonnen. Zum Zeitpunkt dieser Arbeit befinden sich die etwa 40 Module von CSS in unterschiedlichen Stufen der Entwicklung und sind dementsprechend auch unterschiedlich breit implementiert.[39]

Als stabil werden z.B. die Spezifikationen für Selektoren, Namensräume, Farben und „Media Queries“ gesehen, die meisten anderen sind erst als Arbeitsentwurf veröffentlicht worden.

3.2.3 Neue Funktionalitäten

Als neu gelten Möglichkeiten der Bestimmung von Auflösung, Farbtiefe und Seitenverhältnis in den sogenannten „Media Queries“. Mit diesen Informationen können dann Darstellungen auf unterschiedliche Displays zugeschnitten werden.

[39] W3C, CSS – current work

Die Selektoren-Spezifikation sieht umfassendere Möglichkeiten vor, Elemente innerhalb eines Dokuments auszuwählen und anzusprechen. Mit den neuen Möglichkeiten kann z.B. in reinem CSS jedes zweite, dritte oder n-te Element für Formatierungsregeln ausgewählt werden, um z.B. Tabellenreihen unterschiedlich einzufärben.

„CSS Template Layout“ beschreibt eine neue Art Elemente zu positionieren. Hierbei wird die Ausrichtung der Elemente untereinander und nicht im Dokument definiert, um eine höhere Flexibilität zu erreichen.

Obwohl CSS als Formatierungssprache für Texte entwickelt wurde, wird momentan auch an einer Spezifikation für Geräuscheffekte gearbeitet. Die „CSS Aural Style Sheets“ sollen Möglichkeiten bieten, Sound-Effekte zuzuordnen und bezüglich Überlagerung und Positionierung am rechten oder linken Stereobereich zu formatieren. Für diese Spezifikation ist noch kein Arbeitsentwurf veröffentlicht worden und auch kein fester Termin genannt. Somit sind diese Möglichkeiten noch in keinem Browser implementiert und werden wahrscheinlich erst in einigen Jahren die entsprechende Reife erlangen.

Sehr weit entwickelt ist hingegen die Spezifikation für Hintergründe und Rahmen, die es unter anderem nun erlaubt mehrere Hintergründe übereinander zu legen, Hintergrundbilder zu skalieren oder Ecken von Boxen abzurunden.

Formularelementen in verschiedenen Aktivierungszuständen neues Aussehen zuzuordnen ist Teil des „CSS Basic User Interface“. Mit diesen soll eine bessere Integration von Webapplikationen in die Desktopumgebung des Benutzers erreicht werden.

Die Positionierung von Text innerhalb von Texten wird innerhalb des Box-Models beschrieben. Hierfür gibt es zwei Module: das „CSS Basic Box Model“ und das „CSS Extended Box Model“. Mit den dort beschriebenen Eigenschaften soll es z.B. einfacher werden, vertikal verlaufende Sprachen zu formatieren.

Eine ebenfalls oft genannte neue Eigenschaft ist es, Werte von Formatierungen innerhalb des Stylesheets berechnen zu lassen, um genaue Pixelabstände zu erreichen, auch wenn unterschiedliche Formate für Schriftgrößen verwendet werden.

Ebenfalls wichtig ist die Einführung der Unterstützung verschiedener CSS-Profile, namentlich „CSS Mobile Profile“, „CSS Print Profile“ und „CSS TV Profile“, die eine verbesserte Gestaltung für das Anzeigen on HTML-Dokumenten auf mobilen Geräten, für Printausgaben oder innerhalb eines Fernsehers bieten sollen.

Wichtig im Kontext dieser Arbeit sind aber vor allem die Module „CSS Transitions“ und „CSS Animations, sowie „CSS 2D Tranformations“ und „CSS 3D Tranformations“. Wobei die ersten beiden eine JavaScript unabhängige Animation von Elementen zulassen und die letzten beiden verschieben Skalieren und Rotieren im zwei- oder dreidimensionalen Raum ermöglichen. Kombiniert man die Animationsmöglichkeiten mit den Transformationsmöglichkeiten, können so Bewegungen erzeugt werden. Der Unterschied des Transition-Moduls zum Animation-Modul besteht darin, dass mit Transitionen vor allem Übergänge zwischen Darstellungszuständen gemeint sind. Beispielsweise der Übergang in und aus dem Zustand „hover“, der ein mit der Maus überfahrendes Element beschreibt.

Des Weiteren sind Vererbungsdirektiven innerhalb von Stylesheets, Formatierungen für Fußnoten und Querverweise, Direktiven für Eingefügte Inhalte, Hyperlinkformatierungen, Formatierungen innerhalb von Textzeilen und Listen, Formatierungsdirektiven für Mathematische Formeln und Symbole, Farbkodierung und Schriftarten vorgesehen. Zusätzlich gibt es Mehrspalten-Layouts, Namensräume für Formatierungsregeln, Objektmodelle für Skriptinteraktion mit den Formatierungen, Ansichtsfensterpositionierung, eine neue Art der Positionierung von Elementen, Präsentationsebenen, Rasterpositionierungen, Direktiven für Sprachsyntheseprogrammen, spezielle Tabellenformatierungen und Laufschriften.[40]

3.2.4 Für Spiele relevante multimediale Schnittstellen

Da die Audiomöglichkeiten von CSS3 noch nicht ausreichend spezifiziert sind, kann in dieser Arbeit keine Aussage über ihre Verwendbarkeit gemacht werden.

Alle oben genannten Techniken können wichtig für Browserspiele sein, falls diese Spiele bei der Gestaltung der grafischen Benutzeroberfläche auf CSS setzten. Für das Erreichen von multimedialen Effekten, wie Animationen, sind jedoch nur wenige CSS-Eigenschaften von Belang.

Die schon mit CSS2.1 eingeführten Positionierungs- und Abstandseigenschaften sind weiterhin die wichtigsten Grundlagen für Spieleumsetzungen mit Hilfe von CSS. Diese können, ähnlich wie die Skalierung von Elementen, innerhalb von JavaScript-Skripten dynamisch verändert werden, um z.B. Bewegung zu simulieren. Dazu muss alle paar Millisekunden die Position um wenige Pixel von dem Skript verändert werden. Diese Techniken werden z.B. intensiv von Browserspielen verwendet die auf HTML4 Techniken aufbauen.

[40] Vgl. W3C, CSS – current work

Durch die Animationsmöglichkeiten von CSS3, genauer gesagt Animationen und Transitionen, können Elemente auch ohne JavaScript animiert werden. Um Animationen zu nutzen, ist die Definition von sogenannten „Keyframes“ nötig. An diesen orientiert sich der Browser, um daraus eine Animation zu generieren. Bei den „Transitions“ muss das Element, dessen Übergang animiert werden soll, mit entsprechenden Direktiven vorbereitet werden, sowie der neue Zustand selbst definiert werden.

Diese Möglichkeiten bieten nur eine begrenzte zeitliche Kontrolle der Animationen. Um mehrere Animationen in den korrekten Zeitablauf hintereinander zu bringen, wird weiterhin JavaScript benötigt.

3.3 Gegenüberstellung der Standards

In diesem Abschnitt soll eine Empfehlung abgegeben werden, auf welche Technik Browserspiele hauptsächlich aufsetzten sollten. Hierzu muss gesagt werden, dass eine allgemeingültige Empfehlung nicht gegeben werden kann, da jedes Spiel andere Anforderungen an die eingesetzte Technik hat. Es wird daher eine Erläuterung gegeben, was mit den Techniken genau umgesetzt werden kann, wo die Vor- und Nachteile in dem jeweiligen Vorgehen liegen und welche Vorgehensweise für eine prototypische Implementierung gewählt wurde.

Prinzipiell spricht nichts gegen eine Vermischung von CSS3 und HTML5-Techniken, jedoch sollten für eine effiziente Entwicklung von Spielen nicht gleichzeitig zwei Möglichkeiten der Umsetzung für ein identisches Endergebnis gepflegt werden.

Grundsätzlich lässt sich eine Spielwelt mit den hier vorgestellten Techniken auf zweierlei Arten innerhalb eines Browserfensters konstruieren. Die eine Art nutzt HTML-Elemente selbst als Spieleelemente und erreicht Bewegung und Animation durch Modifizierung der CSS-Stile dieser Elemente. Die andere Art erzeugt alle Spielobjekte allein im Zwischenspeicher und nutzt das HTML5-Element „canvas“ als Zeichenfläche, um diese dann darzustellen. Mehrere Darstellungsebenen würden bei dem ersten Verfahren durch die Möglichkeit der Tiefenpositionierung von CSS übereinandergelegt, bei dem zweiten Verfahren ist die Reihenfolge der Zeichenoperationen wichtig, d.h. es muss beispielsweise zuerst der Hintergrund gemalt werden dann die Spielobjekte und dann ggf. ein Vordergrundobjekt.

3.3.1 Vorteile und Nachteile

Während die CSS-basierende Herangehensweise den Vorteil hat, relativ einfach und für Webentwickler leicht verständlich zu sein, hat sie den Nachteil, dass unvorhersehbare Effekte beim Einbetten von CSS3-Spielen in weitere Seiten

eintreten können. Diese resultieren aus dem Umstand, dass die Vererbung von CSS-Stilen sich auf die ganze Seite auswirken kann, falls bei der Selektion der Spiele-Objekte zu unsauber gearbeitet wurde. Ein weiterer Vorteil hingegen ist, dass die Interaktionsmechanismen der HTML-Elemente selbst genutzt werden können, und somit bei Ereignissen, wie dem Anklicken von Spielelementen, diese direkt den Spielelementen zuzuordnen sind. Die Spezifikation der zugrundeliegenden Technik kann sowohl Vorteile, als auch Nachteile bringen. So sind die zu verwendenden CSS2.1-Funktionen lange bekannt und somit auch in nahezu allen Browsern implementiert. Diese hohe Komptabilität würde durch den Einsatz von CSS3-Techniken sofort zunichte gemacht, da hier jeder Browserhersteller unterschiedliche CSS-Spezifikationen bevorzugt implementiert.

Die Lösung, die stark auf JavaScript setzt und ein Canvas-Element zur Darstellung verwendet, kann bei der Einbettung in eine Seite zu weit weniger Nebeneffekten führen. Sie ist flexibler bei Modifikationen, setzt jedoch auch einen gewissen Grad an Know-How voraus. Ein Programmiereinsteiger kann mit einem komplexen Spiel mit dieser Technik schnell überfordert sein, falls er bei der Vorarbeit nicht genügend Sorgfalt in die Qualität des JavaScript-Codes gesteckt hat. Durch die vielfältigeren Programmiermöglichkeiten bei dieser Lösung lassen sich jedoch auch Techniken wie Doppelpufferung und grafisch aufwändigere Systeme, wie etwa Partikelsysteme umsetzten. Doppelpufferung ist hierbei ein Konzept, bei dem ein Resultatbild zuerst im Speicher zusammengesetzt wird, bevor es als Ganzes auf den Bildschirm gemalt werden kann. Durch diese Technik wird oft ein flimmerfreieres bewegtes Bild erreicht. Eine höhere Bildqualität wird mit dem Canvas-Element auch dadurch erreicht, dass es hier möglich ist „zwischen“ den Pixeln zu zeichnen, also Positionsangaben von Elementen nach dem Komma zu verwenden. Der Browser rechnet die dadurch entstehenden Ungenauigkeiten möglichst genau um. Derartige Möglichkeiten sind mit einer CSS-Umsetzung nicht gegeben. Die Spezifikation des HTML5-Canvas-Elementes ist entweder implementiert oder nicht, hier gibt es keinen Grad der Unsicherheit, wie es bei verschieden CSS3-Techniken der Fall wäre. HTML5-Canvas Spiele laufen in jedem Browser, der diese eine Technik unterstützt.

Bei der clientseitigen Ausführgeschwindigkeit liegen CSS-basierende Lösungen vor denen mit dem HTML5-Canvas.[41] Dieser Vorteil wird jedoch durch einen höheren Hauptspeicherverbrauch erkauft. Bei Szenarien mit vielen Spielobjekten

[41] Kistner, Test auf phrogz.net

wird allerdings die Canvas-Lösung schneller ausgeführt.[42] Dieser Umstand ist durch die Eigenschaft des Canvas zu erklären, Grafikobjekte nach einem Zeichenvorgang nicht einzeln im Speicher behalten zu müssen.

Eine mögliche Zwischenform wäre die Nutzung von mehreren Canvas-Elementen neben normalen HTML-Elementen, die mit CSS-formatierung übereinander gelegt werden. So kann beispielsweise ein Canvas-Element den Hintergrund übernehmen und die HTML-Elemente davor mit CSS-Formatierungen bewegt werden. Dies hätte jedoch weiterhin das Problem der schlechten Abkapselung vom Rest der Webseite und führt zu einem noch höheren Komplexitätsgrad, da dem Entwickler sowohl die CSS- als auch die Canvas-Techniken vertraut sein müssen.

3.3.2 Entscheidung für eine beispielhafte Implementierung

Die Entscheidung für eine der Techniken hängt von mehreren Faktoren ab.

Ein Faktor ist, ob das Spiel die Flexibilität des Canvas-Elementes auch benötigt. Relativ statische Spiele, mit wenigen Effekten, sind leichter ohne eine Fixierung auf dieses HTML5-Element umzusetzen. Für Spiele deren Aufbau sich nicht schnell ändern und deren Anforderungen klar definiert sind, gilt ähnliches. Das Canvas-Element kann einfache grafische Primitive wie Rechtecke und Kreise selbst zeichnen, CSS-kann dies nicht. Dadurch muss in CSS-Spielen jedes Element als Grafik vorbereitet werden.

Ein weiterer Faktor ist das Zielmedium. Falls das Spiel in eine schon bestehende Seite eingebaut werden soll, so fällt dies mit Canvas-Elementen leichter, wird eine eigene Seite für das Spiel zusammengestellt ist dies weniger problematisch. Soll diese Seite dann auch mit älteren Browserversionen funktionieren, muss von HTML5 und auch CSS3-Techniken abgeraten werden. CSS2.1-Techniken wären in diesem Fall die beste Wahl.

Oft sind der entscheidenden Faktoren die zu Verfügung stehenden Ressourcen und Entwickler. Auf CSS-basierende Spiele sind sicherlich leichter von erfahrenden Webentwicklern umzusetzen, als Canvas-Spiele, deren Grundlagen eher an die grafischen Schnittstellen von klassischen Programmiersprachen wie Java oder C++ erinnern. Ein CSS-Spiel ist auch leichter auf Arbeitspakete zu zerlegen, da es hier weniger Abhängigkeiten der interagierenden Elemente gibt. JavaScript-Programmierung, CSS-Programmierung und Grafikentwicklung halten sich ungefähr in der Waage. Canvas-Spiele sind deutlich mehr von JavaScript

[42] Eigener Test mit angepassten Versionen des Tests von Kistner. (Bei 2000 anstelle von 20 Objekten zeigt der Canvas-Renderer eine doppelt so hohe Framerate im Vergleich zum CSS-Renderer)

abhängig und haben deshalb eine ungleiche Arbeitslastverteilung, falls eine klassische Rollenverteilung in Grafiker, Programmierer etc. vorgenommen wurde.

Für eine prototypische Implementierung innerhalb dieser Arbeit benötige ich ausreichend Flexibilität, da sich an einem Prototyp schnell und oft wesentliche Dinge ändern. Des Weiteren ist eine ausreichend robuste Abkapselung des Projekts wünschenswert, um leicht aus diesem Prototyp verschiedene Spiele für verschiedene Seiten zu entwickeln. Daher entscheide ich mich für das nächste Kapitel dieser Arbeit für ein Spiel, das hauptsächlich das HTML5-Canvas-Element benutzt.

3.4 Technische Aspekte von Spielen basierend auf HTML5 oder CSS3

Beide Spieltypen benötigen JavaScript als Grundlage. Das gilt für auf Canvas-Elementen basierende Spiele in besonders hohem Maße. JavaScript unterliegt entwicklungsbedingt besonderen Einschränkungen, hat aber auch besondere Vorteile. Um dies zu erläutern, wird hier noch einmal auf JavaScript im Detail eingegangen werden.

3.4.1 JavaScript als Programmiersprache

JavaScript, als Möglichkeit HTML-Seiten clientseitig nach dem Aufruf der Seiten zu verändern, gibt es bereits seit Mitte der 90er Jahre. Den Namen JavaScript erhielt die Skriptsprache, die vorher als LiveScript bekannt war, aus marketingtechnischen Gründen, da die Syntax an die Programmiersprache Java erinnerte.

Merkmale der Programmiersprache JavaScript sind dynamische Typisierung, d.h. Variablen werden erst nach einer Wertzuweisung einem Typ zugewiesen und lassen sich während des Programmablaufs in andere Typen abändern. Dazu kommt Objektorientierung, wobei hierbei das Klassenkonzept nicht unterstützt wird. Programmierparadigmen aus der objektorientierten Programmierung lassen sich durch die Verwendung von Prototypen realisieren. Prototypen erlauben es beliebige Objekte als „Vorbild“ für weitere Objekte zu verwenden, die neu erstellten Objekte haben alle Eigenschaften und Funktionen des Vorbildobjekts. Dieses Vorbildobjekt kann jederzeit dynamisch erweitert werden, die Erweiterungen übertragen sich dabei auf alle Objekte, die dieses Objekt zum Vorbild haben. Datenobjekte werden in JavaScript als assoziative Arrays angelegt, die als Schlüssel-Wert-Paare zu verstehen sind. In HTML-Seiten eingebettete JavaScripts werden in Browsern interpretiert und dann ausgeführt,

wobei modernere Browser beschleunigende Techniken wie „trace trees“ oder Just-In-Time-Kompilierung anwenden. [43]

Anders als andere Programmiersprachen befindet sich JavaScript bei der Ausführung im Browser jederzeit in einer sogenannten „Sandbox“. Das Sandbox-Prinzip beschreibt in der Informatik ein Betriebsmodus, in dem Programme vom Rest des ausführenden Systems abgeschirmt werden. Somit ist JavaScript nicht nur auf die Objekte, die der Browser bereitstellt begrenzt. Durch die in allen Browsern vorhandene „Same-Origin-Policy“, einem weiteren Sicherheitskonzept, kann JavaScript-Code nur auf Objekte zugreifen, die von derselben Domain stammen. Zugriffsport und Protokoll müssen ebenfalls identisch sein.

3.4.2 Das Document Object Model (DOM)

Um JavaScript den Zugriff auf alle Objekte einer Seite zu gewähren, wird vom Browser eine geeignete Schnittstelle bereitgestellt. Das Document Object Model ist hierfür die vom W3C standardisierte Schnittstelle. Mit ihrer Hilfe sind lesende und schreibende Zugriffe auf das HTML-Dokument und Stylesheets im Hauptspeicher, sowie die Behandlung von Ereignissen, resultierend aus Benutzerinteraktion oder Veränderungen, möglich. Die Struktur, die für lesenden und schreibenden Zugriff verwendet wird, ist die eines Stammbaums, bei dem jedes HTML-Element einen Knoten in diesem Baum darstellt. Das in einem HTML-Dokument alles umschließende „HTML“-Element ist der Wurzelknoten. Jeder Knoten kann dabei als Kindknoten des Dokuments selektiert werden und hat die Eigenschaften „firstChild“, „lastChild“, „nextSibling“, „previousSibling“, sowie „parentNode“. Diese stehen jeweils für Verweise auf den ersten Kinderknoten, den letzten Kinderknoten, den nächsten Geschwisterknoten, den vorherigen Geschwisterknoten, sowie den Elternknoten und dienen dem Zugriff auf andere Knoten. Zusätzlich bieten die Eigenschaften „nodeName“ und „nodeType“ Zugriff auf das HTML-Element als Zeichenkette, sowie eine Unterscheidung zwischen HTML-Elementknoten, Attributknoten und Textknoten.

3.4.3 Möglichkeiten

JavaScript kann durch das DOM jeden Knoten selektieren, die dazugehörigen Attribute ändern, sowie Knoten erzeugen, anhängen, klonen, löschen und austauschen. Es kann aber auch beliebige Funktionen an die Ereignisse binden, die durch das DOM mitteilt werden. Zusätzlich können HTML-Elemente selbst noch spezielle Methoden und Ereignisse über das DOM bereitstellen. Der Grafik-Kontext des Canvas-Elements kann hier als Beispiel genannt werden. Als weiteres

[43] Wenz (2007)

wichtiges DOM-Objekt lassen sich die XML-Http-Requests nennen, die es ermöglichen zusätzliche HTTP-Verbindungen aufzubauen, und Daten zu lesen. Dieser Vorgang wird oft mit dem Begriff AJAX (Asynchronous JavaScript and XML) beschrieben. Darüber hinaus sind auch Zeitfunktionen, wie das wiederholte Ausführen von Funktionen nach einem bestimmten Zeitintervall, oder nach Timeouts möglich. Bemerkenswert ist ebenfalls das zur Verwendung von elementeigenen Methoden ein HTML-Element nicht in den Dokumentenbaum eingehängt werden muss, es genügt eine Variable zuzuweisen, und dieses Objekt dann wie ein Element anzusprechen. Dies erleichtert z.B. später den Umgang mit Audio-Elementen.

Durch diese vielfältigen Möglichkeiten kann trotz der strengen Einschränkungen eine Vielzahl von kreativen und dynamischen Ideen umgesetzt werden.

4 Demonstration der Gestaltungsmöglichkeit anhand eines Adventure-Spiels

4.1 Überblick

Adventure-Spiele werden oft als eigenes Genre innerhalb der Computerspiele gesehen. Typischerweise wird eine Rahmenhandlung durch Interaktion mit der Spielewelt vorangetrieben. Die Interkation kann zum Bespiel aus dem Lösen von Rätseln, dem Sammeln von Gegenständen oder aus Gesprächen mit Spielecharakteren bestehen. Da die Geschichte bei diesen Spielen im Vordergrund steht, kann sich bei der Gestaltung an anderen, auf Handlung basierenden Medien angelehnt werden. Dies erlaubt eine Fächerung möglicher Spielmotive durch Anlehnung an verschiedene Film- oder Literaturgenres. So sind zum Beispiel Adventures möglich, die Kriminalfälle erzählen, den Spieler Abenteuergeschichten erleben lassen oder auch erotische Motive beinhalten.

Die ersten Adventure-Spiele wurden in reiner Textform präsentiert. Diese sogenannten Textadventures beschreiben die Welt, in der sich der Spieler zurechtfindet, nur in schriftlicher Form. Interagiert wird mit der Spielwelt durch das Eingeben von vorgegebenen Kommandobefehlen. Bekannteste Serie dieser Spieleart ist die Zork-Serie, die Anfang der 80er Jahre von der Firma Infocom vertrieben wurde.

Eine Weiterentwicklung dieser Spielform waren die Grafik-Adventures. Die ersten Spiele dieser Art setzten weiterhin auf Kommandoeingaben per Tastatur, gaben jedoch zusätzlich zur Beschreibung der Spielwelt eine veranschaulichende Grafik aus. Diese Form der Spiele ist, in neuem Gewand, bis heute im japanischen Raum sehr verbreitet und beliebt. Sie wird in diesem Bereich auch „Visual Novel Games“, frei übersetzt visuelles Romanspiel, oder „Japanische Adventures“ genannt. 2006 machten diese Spiele 70% der verkauften Computerspiele in Japan aus.[44]

Als sich Mitte der 80er Jahre dann die Maussteuerung bei Heimcomputern verbreitete, zu dieser Zeit wurde z.B. auch das erste Microsoft Windows veröffentlicht, verwendeten diese Steuerung auch die Grafikadventures. Aus ersten alternativen Steuerungsmöglichkeiten, wie dem Anwählen von Kommandos, entwickelte sich das Steuerungskonzept „Point-and-Click“. Bei diesem Konzept werden mit der Maus spezielle Stellen einer grafischen Bedienoberfläche überfahren und mit einem Klick eine Aktion ausgelöst. Oft ist die Darstellung der Spielwelt selbst dabei eine Bedienoberfläche, in der oder mit

[44] animenewsnetwork.com (2006)

deren Hilfe Kommandos erteilt werden können. Oft wurde auch ein Verben-System verwendet, das dem Spieler erlaubte an Gegenständen in der Spielwelt verschiedene Aktionen durch vorheriges Auswählen eines spezifischen Verbs, wie „Ansehen“, „Drücken“ oder „Aufheben“, durchzuführen. Adventures die ein solches Steuerungskonzept vorsehen werden auch „Point-and-Click“-Adventures genannt. Die bekanntesten Spiele dieser Art stammen dabei von der Firma Lucasfilm Games, die Titel wie „Day of the Tentacle“, „Zak McKracken“ und „Monkey Island“ geschaffen hat.

Die Einführung von CD-ROM-Laufwerken für den Heimcomputer begann Anfang der 90er Jahre. Von den Möglichkeiten dieser damals neuen Technik machte die Firma Cyan, später Cyan Worlds, gebrauch. In denen von dieser Firma veröffentlichten Spielen, wie z.B. „Myst“, wurden die höheren Kapazitäten einer CD-ROM im Vergleich zu Disketten genutzt, indem vorgerenderte aufwändige dreidimensionale Computergrafiken für die Spielwelt eingesetzt wurden. Das Spielgeschehen wurde hierbei aus einer Ich-Perspektive erlebt, wobei die grafische Darstellung der Welt auch gleichzeitig Benutzeroberfläche war. Objekte in der Spielwelt konnten direkt angewählt und manipuliert werden. Die Myst-Reihe galt bis zum Jahre 2002 als die erfolgreichste Spielereihe der Computerspielgeschichte. Ab dann übernahm die populäre Computerspielreihe „Die Sims“ diesen Platz. [45]

Trotz der hohen Popularität des Adventures Ende der 80er und Anfang der 90er Jahre, wurde dieses Genre bis Ende der 90er Jahre von professionellen Entwicklern nahezu aufgegeben. Gründe dafür waren die sich ändernde Demographie der Computerspieler, der technische Fortschritt bei anderen Genres, wie bei den Ego-Shootern, und die kommerziellen Misserfolge von Adventures, die versuchten neue Technologien einzuführen.

In den letzten Jahren ist jedoch wieder ein Aufwärtstrend dieses Genres zu sehen. Die sich abermals ändernde Demografie von Computer-und Videospielern und die Möglichkeit Point-and-Click-Adventures auf Spielekonsolen, wie Nintendos Wii oder DS, durch Zeiger und Touchscreen-Technologie sinnvoll spielen zu können, können hier als Anreize für diesen Aufschwung gesehen werden.

Adventures haben auch den Vorteile Lernspiele oder Serious-Games durch eine klare Struktur zu unterstützen. Wissensvermittlung kann narrativ geschehen und durch Rätsel und Gespräche mit Spielecharakteren vertieft werden.

[45] Walker (2002)

Als Browserspiele lassen sich Adventures bereits oft bei den Flash-Spiele finden. Beliebt hierbei sind vor allem die „Escape-Game“-Reihen, bei denen der Spieler versucht aus einem geschlossenen virtuellen Raum zu entfliehen.

4.2 Bestandteile

Eine aktive Entwicklungsgemeinde, die sich damit beschäftigt ältere Point-and-Click-Adventures auf vielen Plattformen lauffähig zu machen, ist die Gruppe die hinter dem SCUMMVM-Projekt steht. Innerhalb dieses Projekts werden verschiedene alte Adventures analysiert und dann neu implementiert. Hierbei werden oft die alten Spiele-Ressourcen aus den originalen Spieledateien verwendet. Die Verknüpfung der Ressourcen zu einem Spiel passiert dann jedoch nicht länger durch den originalen Interpreter des Spiels, sondern durch definierte Skripte der SCUMMVM-Entwickler.

Aus der öffentlich zugänglichen Dokumentation des Projektes kann entnommen werden, aus welchen Teilen sich ein typisches Adventure zusammensetzt. Unabhängig von der Verknüpfung untereinander sind die zugrundeliegenden Ressourcen eines Adventures ähnlich. So besteht die Grafikdarstellung aus den Ressourcen für Akteure, Hintergrundbilder, Schriftarten, Animationen, Objekte und Videos. Die Geräuschkulisse besteht meist aus Musik, Sound-Effekten und Sprachaufnahmen.[46]

Hierbei muss beachtet werden, dass die Ressourcen für Akteure, Animationen und Objekte meist aus sogenannten Sprites bestehen. Der Begriff Sprite beschreibt heutzutage eher generell Grafikobjekte, die vor einem Hintergrund dargestellt werden und sich vor diesem bewegen. In den frühen Zeiten der Computerspiele waren Sprites Grafikobjekte, die direkt von der Grafikhardware über die eigentliche Bildschirmausgabe gelegt werden konnten. Durch die Spritedarstellung für Spielobjekte wurden die knappen Rechenressourcen der ersten Heimcomputer entlastet.

Sprites sind oft in Spritemaps, auch Spritesheets genannt, zusammengefasst. Das sind große Bilddateien, die z.B. alle Einzelbilder für die Animation eines Objekts beinhalten. Um ein Objekt zu animieren, muss jetzt zur Laufzeit des Spiels nur der Ausschnitt von diesem Sheet, der dargestellt wird, verschoben werden, um die Einzelbilder der Animation der Reihe nach abzuspielen. Diese Technik kann man auch mit HTML5 benutzen um die Anzahl der Einzelbilder zu reduzieren, aus denen das Spiel sich zusammensetzt. Dies führt ebenfalls zu schneller geladenen Spielen und leichterem Überblick über die Grafikressourcen.

[46] Vgl. ScummVM Project (2010)

Abbildung 7: Spritesheet der Spielfigur für den Prototyp (Bewegungsanimation)

Anders als im SCUMMVM-Vorbild muss bei einem Browserspiel nicht unbedingt eine Schriftart-Ressource erstellt werden, da auf die Schriftarten des Browsers, die auch in der CSS-Formatierung eingesetzt werden, zugegriffen werden kann.

Die Skripte für die Spielemechanik selbst unterscheiden sich stark von Spiel zu Spiel. Möglich ist es zum Beispiel für jede Spielszene zu definieren, wie sich diese zusammensetzt oder ein generelles Skript für alle Szenen zu definieren. Allen Skripten gemeinsam ist jedoch eine zentrale Schleife, die für jedes Einzelbild der Darstellung durchlaufen wird. Diese wird Gameloop genannt und umfasst das Zeichnen der Spielfläche, dem Anpassen aller für das Spiel relevanten Variablen und der Kontrolle von Bedingungen innerhalb der Spiellogik.

4.3 Umsetzung

Für eine prototypische Umsetzung wende ich den Ansatz an, ein generelles Skript auf JavaScript-Basis zu definieren. Dieses Skript lädt alle benötigten Ressourcen vor, initialisiert die Spielwelt und startet den Gameloop.

Im aufzurufenden HTML-Dokument des Prototyps muss wenig definiert werden. Es wird lediglich die Definition des Canvas-Elements, das als Zeichenfläche für das Spiel verwendet wird, ein Verweis auf die JavaScript-Datei, die das Skript für das Adventure enthält und ein kurzer JavaScript-Aufruf, der nach dem vollständigen Landen des Dokuments das Adventure-Skript startet, benötigt. Alle weiteren Definitionen auf der Seite dienen der Gestaltung des „Rahmens“ der Spielfläche, welcher z.B. mittig ausgerichtet wird.

Um das Adventure-Skript von anderen Skripten der Seite abzukapseln, werden alle Funktionen und Objekte als Kinderelemente eines übergeordneten globalen „Adventure-Objekts“ definiert. Die Ressourcen des Adventures sind in diesem Fall ein Unterobjekt des Adventures und werden zur Vereinfachung in Bilder, Audio-Inhalte und Texte aufgeteilt. Hierbei implementiert jedes Resourcen-Objekt ein Prototyp-Objekt, welches zugrundeliegende Eigenschaften und Funktionen für die Verwendung innerhalb des Spiels bereitstellt. Die

Beschreibung für eine Szene-Objekt nutzt die Resourcen-Objekte um eine Szene zusammenzusetzten. Die dynamische Erweiterbarkeit aller Objekte durch Einsatz von JavaScript ermöglicht hier eine hohe Flexibilität. Möglichkeiten sind zum Beispiel ähnlichen Grundobjekten bei Definition einer Szene verschiedene Aktionen zuzuweisen, oder diese sogar zu Laufzeit zu verändern. Zusätzlich lässt das Verhalten von JavaScript Objekte als assoziative Arrays anzulegen eine einfache Benennung und Adressierung der Ressourcen zu.

Durch die HTML5-Technologie müssen jedoch auch Besonderheiten beachtet werden, die nicht sofort ersichtlich sind.

So haben Audio-Objekte die Eigenschaft nur einen virtuellen Audio-Kanal zur Verfügung zu haben, das gleichzeitige Abspielen derselben Ressource erfordert ein besonderes Vorgehen. Ein Weg diese Eigenschaft zu umgehen ist ein Array an Audio-Objekten, die dynamisch die Ressourcen-Objekte als Quellen benutzten.

Für das wiederholte Aufrufen des Gameloops bieten sich zwei JavaScript Methoden an: setInterval und setTimeout. Die erste Methode ruft eine beliebige Methode nach einem festen Intervall immer wieder auf, die zweite eine beliebige Methode einmalig nach Verstreichen einer angegebenen Zeitspanne. Für den Gameloop drängt sich setInterval in erster Betrachtung durch die einfache Handhabung auf. Dieses Vorgehen hat jedoch einen entscheidenden Nachteil. Anders als setTimeout, wird die aufrufende Funktion bei setInterval nach Aufruf der aufzurufenden Funktion nicht beendet. Dies kann bei Ausführung im Browser dazu führen, dass bei Aufwändigen Berechnungen, die den JavaScript-Interpreter blockieren, sich sehr viele Funktionsaufrufe aufreihen und das Skript nicht mehr reagieren lassen. Die setTimeout-Methode würde bei Blockierung mitblockieren und somit keine weiteren Aufrufe auslösen. Für stabile Spiele muss also die setTimeout-Methode verwendet werden, die am Ende des Gameloop jedes Mal erneut aufgerufen wird. An dieser Stelle wurde beim Prototyp noch eine Berechnung eingefügt, die eine zeitlich möglichst gleichmäßige Ausführung zu gewährleisten. Diese Berechnung reduziert die Zeitspanne der setTimeout-Methode um die Zeit, die der aktuelle Durchlauf des Gameloops benötigte.

Um zu bestimmen über welchem Spielobjekt sich der Mauszeiger befindet und welches angeklickt wurde, wird in der vorliegenden Implementierung ebenfalls ein besonderes Vorgehen gewählt. Da die Darstellung im Canvas-Element nur eine Darstellung der sich im Speicher befindlichen Objekte ist, kann bei einem Klick auf ein Canvas-Objekt der Browser nur einen Klick auf das Element selbst feststellen. Es unterliegt dann der Spielelogik, die Koordinaten des Klicks zu bestimmen und alle Objekte im Speicher auf ihre entsprechende Positionierung zu prüfen. Liegt ein Objekt auf den ermittelten Koordinaten, kann der Klick dem

Objekt zugeordnet werden. Ein Spezialfall tritt bei sich überlappenden Objekten auf. Bei einem Klick auf die überlappende Stelle, wird das zuletzt der Szene hinzugefügte Objekt als Angeklickt betrachtet. Hier wären noch Verfeinerungen wie z.B. Tiefenkoordinaten möglich.

Für den Test, ob sich die Spielfigur zu einem bestimmten Punkt bewegen in einer Szene darf, muss ein ähnliches Vorgehen gewählt werden. Hierbei ist jedoch innerhalb der Spiellogik zu prüfen, ob die errechneten Koordinaten innerhalb eines Polygons liegen, das für die Szene als erlaubte Bewegungsfläche definiert wurde. Ein einfacher Algorithmus basierend auf der Annahme, dass ein Strahl ausgehend von einem Punkt innerhalb eines Polygons eine ungerade Anzahl von Seiten des Polygons schneidet, reicht in diesem Fall aus.

Als letzte Besonderheit ist anzumerken, dass nicht alle Bild-Ressourcen unbedingt aus Bilddateien bestehen müssen. Durch die Möglichkeit von JavaScript Objekte im Speicher zu generieren, ist es auch möglich Canvas-Elemente im Speicher zu generieren. Auf diesen Elementen kann daraufhin gezeichnet werden, um das resultierende Bild als Ressource zu verwenden. Mit dieser „on-the-fly“–Bildgenerierung lassen sich Effekte, einfache visuelle Objekte oder kleinere Animationen direkt im Skript bewerkstelligen.

Der entstandene Prototyp beachtet all die Besonderheiten und implementiert drei einfache Spielszenen mit grundlegender Interaktion. Innerhalb der Szenen kann der Spieler eine Spielfigur durch einfaches Klicken durch einen Raum bewegen. Spielobjekte sind durch einen sich ändernden Mauszeiger beim Überfahren mit der Maus kenntlich gemacht. Nach Anklicken eines Spielobjekts, gibt das Spiel dem Spieler für alle Objekte die Wahl zwischen zwei Aktionen. Die erste Aktion ruft eine die Beschreibung des Objektes ab und lässt diese über der Spielfigur erscheinen. Das damit verbundene Symbol ist die Lupe. Die zweite Aktion ist die dem Objekt hinterlegte Aktion auszuführen. Hierbei bewegt sich die Spielfigur zu dem angesprochenen Objekt hin. Das damit verbundene Symbol sind Zahnräder. Von einer Szene in eine andere kann der Spieler durch benutzten einer Tür gelangen.

Während die erste Szene einen in einem Bildprogramm gezeichneten Hintergrund benutzt, sind die Hintergrundbilder der anderen Szenen per Skript generiert. In diesen Szenen wird auch ein generierter Bildeffekt über die Szenen gelegt.

Ausschnitte dieser Implementierung können dem Anhang entnommen werden. Eine lauffähige Version des Prototyps kann zu jeder Zeit über die Internetadresse http://www.stromfrei.de/chap/ aufgerufen und ausprobiert werden. Auf Quelltextverschleierung der Skripte mit Hilfe diverser JavaScript-Hilfsprogramme wurde in diesem Fall, aus Gründen der Nachvollziehbarkeit, verzichtet.

5 Zusammenfassung und Ausblick

Wie durch den Prototyp und die vorherigen Ausführungen zur Technik belegt ist, ist es leicht möglich Anforderungen umzusetzen, die Browserspiele an eine zugrundeliegende Technik stellen. Für fehlende Funktionalitäten bietet JavaScript genügend Flexibilität um diese nachzubilden. Bekannte Algorithmen für Computerspiele können innerhalb von JavaScript schnell umgesetzt werden, die fehlende „echte“ Objektorientierung von JavaScript ist hierbei kein Hindernis. Viele entstehende und vorhandene Skript-Bibliotheken für Browserspiele aufbauend auf dieser Technik unterstützen diese Annahme.[47]

Somit bieten HTML5 und die damit verbundenen Techniken eine starke neue Möglichkeit für dynamische Inhalte an, die bisher nur von Browser-Plug-Ins umgesetzt werden konnten.

Trotz des frühen Stadiums der HTML5-Spezifikation, bieten die Implementationen der neusten Browsergenerationen eine erstaunlich standardkonforme und stabile Unterstützung der neuen Techniken. Auch wenn einige Browser der neusten Generation nicht alle neuen Techniken umsetzten, können zumindest die Techniken verwendet werden, die der Prototyp implementiert. Für Spiele kann aus dieser Sicht schon fast die Empfehlung gegeben werden, ausschließlich auf die neue Technik zu setzten. Die Vorteile von Browser-Plug-Ins wie Flash, liegen jedoch in der immer noch höheren Ausführgeschwindigkeit, den professionelleren Werkzeugen zur Inhaltserzeugung und der Möglichkeit nicht an die Sandbox des Browsers gebunden zu sein. Durch die Weiterentwicklung der JavaScript-Interpreter von modernen Browsern und das hohe Interesse der Öffentlichkeit an der HTML5-Technologie werden sich diese Effekte immer weiter abschwächen. Die Firma Adobe bietet sogar Tools an, die Flash-Objekte in HTML5 konvertieren[48]. Daher kann angenommen werden, dass die Verbreitung von Flash zurückgehen wird, jedoch keine Technik die Andere komplett ersetzten wird.

Als Ziele weiterer Betrachtungen lassen sich kommende Spezifikationserweiterungen, wie etwa der Zugriff auf Geräte wie Webcams, genauere Markanalysen für den Bereich der Browserspiele oder die Umsetzung weiterer Spielegenres ausmachen. Die Beschäftigung mit diesem jungen und innovativen Themenbereich wird vor allem durch die allgemein hohe Dynamik der Webtechnologien und der Computerspielbranche auch in Zukunft noch lange spannend bleiben.

---

[47] Vgl. impactsjs.com

[48] Adobe Inc. (2010)

6 Literaturverzeichnis

Adobe Systems, Flash Player Version Penetration. Abgerufen am 11. 3. 2011 von http://www.adobe.com/products/player_census/flashplayer/version_penetration.html

Adobe Systems. (28. 10 2010). *Adobe Blog:* Adobe demos Flash-to-HTML5 conversion tool. Abgerufen am 18. 3. 2011 von http://blogs.adobe.com/jnack/2010/10/adobe-demos-flash-to-html5-conversion-tool.html

animenewsnetwork.com. (8. 2 2006). *AMN and Anime Advanced Announce Anime Game Demo Downloads*. Abgerufen am 15. 3. 2011 von http://www.animenewsnetwork.com/press-release/2006-02-08/amn-and-anime-advanced-announce-anime-game-demo-downloads

Apperley, T. (2006). Genre and game studies: Towards a critical approach to videogame genres. *Simulation & Gaming: An International Journal of Theory Practice and Research 37(1),* S. 6-23.

Baumgärtel, T. (10. 11 1998). *Interview mit Nolan Bushnell, Erfinder von Pong und Atari-Gründer*. Abgerufen am 11. März 2011 von http://www.heise.de/tp/r4/artikel/2/2525/1.html

Bigpoint GmbH. Abgerufen am 11. März 2011 von http://www.bigpoint.net/index.es?action=products&subpage=products_games

BIU e.V. (2010). *Spielgewohnheiten im Internet: Die Nutzung von Online- und Browser-Games in Deutschland – Repräsentative Befragungen in 2008, 2009 und 2010.* Berlin.

Bjørstad, T. E., *The Jentonic Mirror Tools -- Sol Growth*. Abgerufen am 2011. 3 18 von http://www.student.informatik.tu-darmstadt.de/~misar/psa/history/stats.htm

Blow, J. (15. 2 2011). Do you believe social games are evil? “Yes. Absolutely.”. (PCGamer.com, Interviewer)

British Museum - The Royal Game of Ur. Abgerufen am 2011. März 10 von http://www.britishmuseum.org/explore/highlights/highlight_objects/me/t/the_royal_game_of_ur.aspx

browsergames.de. (19. 1. 2011). *Poisonville: Teuerstes Browserspiel der Geschichte eingestampft - Die Fakten, die Hintergründe*. Abgerufen am 3. 3. 2011 von http://poisonville.browsergames.de/news/1/2295-poisonville-teuerstes-browserspiel-der-geschichte-eingestampft-die-fakten-die-hintergruende.html

Bundesverband Digitale Wirtschaft e.V. (2010). *Online Gamesreport 2010, Studie der Fachgruppe Connected Games im BVDW.* Düsseldorf.

Crawford, C. (1982). *A Taxonomy of Computer Games.* Washington State University.

e.V., B. I. (2011). *Marktzahlen Computer- und Viedeospiele Gesamtjahr 2010.* Berlin: www.biu-online.de.

European Commission. (2004). *European Interoperability Framework for pan-European eGovernment Services, Version 1.0* . Luxembourg.

Frey. (2004). Spiele mit dem Computer Fantasy, Science Fiction, Rollenspiele & Co. Ein Reiseführer. Kirchberg: Smartbooks.

Gamesconvention.com. *Gamesconvention Online*. Von http://www.gamesconvention.com abgerufen

Justus, C., & Schultheiss, D. Geschäfts-und Erlösmodelle von Browsergames. *Making Games Magazin 02/2010*, S. 55.

Kistner, G., *phrogz.net:* Moving Image via Canvas vs CSS. Abgerufen am 10. 3 2011 von http://phrogz.net/tmp/image_move_speed.html

Magdans, F. (2008). Game Generations: Spieleentwickler und Szenekenner erzählen, wie Computer- und Videospiele unsere Freunde wurden . Marburg: Schüren Verlag.

Making Games Magazine. Online 2015 Im Gespräch mit Heiko Hubertz und Ralf Adam. *Making Games Magazine 05 /2010*.

Medienboard Berlin-Brandburg GmbH. (2008). *Games - Informationen zum Medienstandort Berlin-Brandenburg.* Potsdam.

Onlive. *Onlive is Cloud Gaming*. Abgerufen am 2011. März 11 von http://www.onlive.com/service/cloudgaming

Pilgrim, M. (2010). *HTML5 Up and running.* Sebastopol: O'Reilly Media Inc. .

Riastats. Abgerufen am 12. 3 2011 von http://riastats.com/

Runescape. Abgerufen am 13. 3 2011 von http://www.runescape.com

Schmidt, J., Dreyer, S., & Lampert, C. (2008). *Spielen im Netz Zur Systematisierung des Phänomens "Online-Games"*. Hamburg: Verlag Hans-Bredow-Institut.

Schultheiss, D. (2009). I*m Reich der interstellaren Händler: Internetgames als innovative Spielform: Eine Längsschnittstudie zu Spielmotivationen, Spielerleben und Spielverhalten am Beispiel eines Langzeit-Browsergames* . Ilmenau : TU Ilmenau Universitätsbibliothek.

ScummVM Project. (15. 9 2010). *About - ScummVM The inner workings of adventure games*. Abgerufen am 18. 3 2011 von: http://wiki.scummvm.org/index.php/About#The_inner_workings_of_adventure_games

SelfHTM, HTML-Dateien selbst erstellen. Abgerufen am 11. 03 2011 von: http://de.selfhtml.org/

The IGDA Casual Games SIG. (2006). *2006 Casual Games White Paper*.

Unity Technologies, *Fast Facts*. Von http://unity3d.com/company/fast-facts

USA Today: "For Social Networks, it's Game On". Abgerufen am 10. 3 2011 von http://www.usatoday.com/tech/gaming/2009-10-15-games-hit-social-networks_N.htm

W3C. (14. 2 2011). *W3C Confirms May 2011 for HTML5 Last Call, Targets 2014 for HTML5 Standard*. Abgerufen am 12. 3 2011 von http://www.w3.org/2011/02/htmlwg-pr.html

W3C. *CSS - aktuelle Arbeit & Hinweise wie Sie daran teilhaben können*. Von http://www.w3.org/Style/CSS/current-work abgerufen

Walker, T. (22. 3 2002). *gamespot.com: The Sims overtakes Myst*. Abgerufen am 9.3.2011 von: http://www.gamespot.com/pc/strategy/simslivinlarge/news_2857556.html

Wenz, C. (2007). *JavaScript und AJAX , Das umfassende Handbuch* . Galileo Computing.

Zynga. *zynga.com*. Von http://www.zynga.com/ abgerufen

Anhang A: HTML-Dokument index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Canvas HTML5  Adventure Prototype</title>
    <style type="text/css">
      #main_canvas {
      /* mittige Positionierung */
       border: 2px solid black;
       position:relative;
       left:50%;
       top: 20px;
       padding : 5px;
       margin-left: -400px;
       /* CSS3-Feature : Abgerundete Ecken */
       -moz-border-radius: 8px 8px 8px 8px;
       border-radius:  8px 8px 8px 8px;
       }
       body{
       background-color : #333;
       margin:0px;
       border:0px;
       }
    </style>
  </head>
  <body >
    <canvas id="main_canvas" width="800" height="480"></canvas>
    <script type "text/javascript" src="adventure.js"></script>
    <script type="text/javascript">
        window.onload = function(){
            Adventure.init("main_canvas");
        };
    </script>
  </body>
</html>
```

Anhang B: JavaScript-Datei adventure.js Ausschnitt: Gameloop

```
gameloop : function (){
            Adventure.draw();
            Adventure["actor"].update();
            if(Adventure["gameVariables"]["actionBound"]&&
Adventure.distanceToActor(Adventure["gameVariables"]["boundedObjec
t"]) <  40){
      Adventure["gameVariables"]["actionBound"] = false;
      Adventure["gameVariables"].boundedObject.action();
            }
var time_delta =
 (new Date().getTime()) - Adventure["gameVariables"]["timer"];
Adventure["gameVariables"]["timer"]+= time_delta;
Adventure["actor"]["timeDelta"]+= time_delta;
if(Adventure["gameVariables"]["text"]["textPresent"]){
                  Adventure["gameVariables"]["text"]["timer"]+=
time_delta;
if(Adventure["gameVariables"]["text"]["timer"]                    >=
Adventure["gameVariables"]["text"]["timeout"]){
      Adventure["gameVariables"]["text"]["paging"]++;
      Adventure["gameVariables"]["text"]["timer"] = 0;
if(Adventure["gameVariables"]["text"]["paging"]                   >=
Adventure["gameVariables"]["text"]["text"].length)
{
Adventure["gameVariables"]["text"]["paging"] = 0;
Adventure["gameVariables"]["text"]["textPresent"] = false;
}
}
}
time_delta -= (1000/Adventure["gameVariables"]["frameRate"]);
if(Adventure["gameVariables"]["focus"]){
var waitingTime = ((1000/Adventure["gameVariables"]["frameRate"])-
time_delta);
Adventure["gameVariables"]["timeout"]=
window.setTimeout(Adventure.gameloop,waitingTime);
}
}
```

Anhang C: JavaScript-Datei adventure.js Ausschnitt: playSound

```
playSound: function(s){
            for (a=0;a<Adventure["audioChannels"].length;a++) {
                  var thistime = new Date();
                  if   (Adventure["audioChannels"][a]['finishTime']
< thistime.getTime()) {
      Adventure["audioChannels"][a]['finishTime']                 =
thistime.getTime() + s.duration*1000;

      Adventure["audioChannels"][a]['audio']['src'] = s['src'];

      Adventure["audioChannels"][a]['audio'].load();

      Adventure["audioChannels"][a]['audio'].play();
                        return a;
                  }
            }

      }
```

Anhang D: JavaScript-Datei adventure.js Ausschnitt: Draw

```
draw : function (){
            var c = Adventure["buffer"]["context"];
            var                    activeScene                    =
Adventure["scenes"][(Adventure["gameVariables"]["activeScene"])];
            c.drawImage(activeScene["backgroundImage"],0,0);
            var sceneObjects = activeScene["objects"];
            for( var o = 0;o  <sceneObjects.length;o++){
            c.drawImage(sceneObjects[o].image,
            sceneObjects[o].imageOffsetX,
            sceneObjects[o].imageOffsetY,
            sceneObjects[o].width,
            sceneObjects[o].height,
            sceneObjects[o].posX,
            sceneObjects[o].posY,
            sceneObjects[o].width,
            sceneObjects[o].height);
            }
            var a = Adventure["actor"];
            c.drawImage(a["image"],
            a["imageOffsetX"],
            Adventure["actor"]["heading"]                          *
            a["imageOffsetY"],a["width"],a["height"],a["posX"]-
            a["width"]/2,a["posY"]-
            a["height"],a["width"],a["height"]);

            for(var f in activeScene["additionalRenderSteps"]){
             activeScene["additionalRenderSteps"][f](c);
            }
            if(Adventure["gameVariables"]["tools"]["active"]){

      c.drawImage(Adventure["resources"]["images"]["iconsheet"]["i
mage"],0,0,50,50,Adventure["gameVariables"]["tools"]["x"],Adventur
e["gameVariables"]["tools"]["y"]+50,50,50);

      c.drawImage(Adventure["resources"]["images"]["iconsheet"]["i
mage"],50,0,50,50,Adventure["gameVariables"]["tools"]["x"],Adventu
re["gameVariables"]["tools"]["y"],50,50);
            }
            c.fillStyle = '#000';
      if(Adventure["gameVariables"]["text"]["textPresent"]){
      c.fillText(Adventure["gameVariables"]["text"]["text"][(Adven
ture["gameVariables"]["text"]["paging"])],a["posX"],a["posY"]-
a["height"]);
            }
Adventure["canvas"]["context"].drawImage(Adventure.buffer,0,0);

}
```

Anhang E: JavaScript-Datei adventure.js Ausschnitt: on-the-fly: Background

```
"blueroombg" : {
      load: function(){
      Adventure.resources.canvasImage.apply(this,[Adventure["canva
s"].width+200,Adventure["canvas"].height+200]);
      var c = this["c"];
      c.translate(-120,-120);
      c.rotate(14* Math.PI / 360);
      var                              grad                              =
c.createLinearGradient(0,0,Adventure["canvas"].width+200,Adventure
["canvas"].height+200);
      grad.addColorStop(0, '#000');
      grad.addColorStop(1, '#8888ff');
      c.fillStyle = grad;
      c.fillRect(0,0,Adventure["canvas"].width+200,Adventure["canv
as"].height+200);
      c.strokeStyle = '#ffffff';
      c.beginPath();
      for(var t=0;t<= (Adventure["canvas"].width+200)/20; t++){
            c.moveTo( t*20   , 0);
            c.lineTo( t*20   , Adventure["canvas"].height+200);
                              }
      for(var t=0;t<= (Adventure["canvas"].height+200)/20; t++){
            c.moveTo(0,              t*20                              );
            c.lineTo(Adventure["canvas"].width+200, t*20   );
                              }
            c.closePath();
            c.stroke();
                  }

                  }
```

Versicherung

Ich versichere, dass ich diese Bachelorarbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt, nur die angegebenen Quellen benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Dortmund, den .....