Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 2807

Mobile ortsbasierte Browserspiele

Christoph Stach

Studiengang:	Informatik
Prüfer:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer:	Dipl.-Inf. Andreas Brodt
begonnen am:	14. Juli 2008
beendet am:	13. Januar 2009
CR-Klassifikation:	H.5.1, H.5.2, K.8

Kurzfassung

Das Spiel liegt in der Natur der menschlichen Psyche - sei es, um das Unbekannte zu erforschen, um andere nachzuahmen, um soziale Beziehungen zu pflegen oder aus Langeweile. Während früher Spiele in der realen Welt stattgefunden haben, so gewannen Computerspiele mit der stärkeren Verbreitung von Heimcomputern zu Beginn der 80er Jahre des 20. Jahrhunderts an Bedeutung. Mit dem Erscheinen und dem großen kommerziellen Erfolg von Handheld-Konsolen wie beispielsweise dem Game Boy wuchs das Interesse an mobilen Spielen.

Die heutigen verfügbaren mobilen Computer mit Industriestandards, wie GPS oder Bluetooth, erlauben es dem Nutzer nicht nur jederzeit und überall Zugriff auf seine Spiele zu haben, sondern man kann zusätzlich die Umgebung des Nutzers in die virtuelle Welt einfließen lassen. Die sogenannten „Pervasive Games“ nutzen diese Techniken z.B. indem der Avatar nicht über ein Eingabegerät gesteuert wird, sondern durch eine physische Positionsänderung des Users.

Die vorliegende Diplomarbeit versucht einen Überblick über die bislang bestehenden mobilen ortsbasierten Spiele und Spielkonzepte zu geben. Des Weiteren setzt sie sich kritisch damit auseinander, in wie fern sich diese für das Nokia N810 Internet Tablet als Browserspiele umsetzten lassen. Dazu wird erörtert, welche Anforderungen an mobile ortsbasierte Browserspiele gestellt werden. In diesem Zusammenhang werden nicht nur benötigte technische Ressourcen betrachtet, sondern auch die Voraussetzungen, um einen möglichst hohes Spielvergnügen zu garantieren. Anhand dieser Anforderungen wurde ein Framework erstellt, mit dessen Hilfe ebenfalls ein exemplarischer Prototyp eines mobilen ortsbasierten Browserspieles implementiert wurde. Abschließend wurde dieser Prototyp anhand der im Vorfeld aufgestellten Anforderungen kritisch analysiert und bewertet.

Abstract

Playing games lies within men's nature - may it be to explore the unknown, to imitate the others, to network social relationships or against boredom. While in former times games took place in the real World, computer games gained in importance through the strong spread of home computers at the beginning of the 20th century's 80s. By the emerge and huge commercial success of handheld-consoles like the game boy interest in mobile games rose.

Nowadays available mobile computers featuring industry standards like GPS or Bluetooth give the user not only the possibility having access to his games anytime and anywhere, but it can also integrate the user's environment in the virtual world. The so called „pervasive games" use these technics e.g. by navigating the avatar not by a controller but by changing the user's physical position.

The thesis at hand overviews up to now existing mobile location based games and gaming concepts. In addition it censors whether those games can be converted into browser games running on the Nokia N810 Internet Tablet. For this reason it will be argued which requirements are defined against mobile location based browser games. In this coherence not only used technical resources are examined but also the premises to assure a game's enjoyment as high as possible. A framework was build on the basis of these requirements and a mobile location based browser game's prototype was implemented with its help. Finally, this prototype was critically analyzed and evaluated against the set up requirements in the beginning.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
AR	Augmented Reality
AS	ActionScript
BMI	Body-Mass-Index
CSS	Cascading Style Sheets
CYSMN	Can You See Me Now?
DAO	Data Access Objects
DBMS	database management system
DCCI	Delivery Context: Client Interfaces
DOM	Document Object Model
EDSAC	Electronic Delay Storage Automatic Calculator
FSM	finite state machine
GBA	Game Boy Advance
Gilead	Generic Light Entity Adapter
GPS	Global Positioning System
GUI	Graphical User Interface
gwt	google web toolkit
HMD	Head Mounted Display
HQL	Hibernate Query Language
HTML	Hypertext Markup Language
IEEE	Institute of Electrical and Electronics Engineers
IPerG	Integrated Project of Pervasive Games
IR	Infrarot
ISO	Internationale Organisation für Normung
JDK	Java Development Kit
JEB	Java Enterprise Beans
JSP	JavaServer Pages
KI	Künstliche Intelligenz
MOPET	Mobile Personal Trainer
MP	Multiplayer
OSI-Modell	Open Systems Interconnection Reference Model
PDA	Personal Digital Assistant
PHP	PHP: Hypertext Preprocessor
POJO	Plain Old Java Object

PS3 ..... PlayStation 3
RFID ..... Radio Frequency Identification
RPC ..... Remote Procedure Call
SLOC ..... Source lines of code
SMIL ..... Synchronized Multimedia Integration Language
SP ..... Singleplayer
SVG ..... Scalable Vector Graphics
TEA ..... Technology for Enabling Awareness
UMTS ..... Universal Mobile Telecommunication System
VR ..... Virtual Reality
W3C ..... World Wide Web Consortium
WLAN ..... Wireless Local Area Network
WML ..... Website Meta Language
WPA ..... Wi-Fi Protected Access
XHTML ..... Extensible Hypertext Markup Language
XML ..... Extensible Markup Language
XSLT ..... Extensible Stylesheet Language Transformation

Kapitel 1

Einleitung

Hoher Sinn liegt oft in kind'schem Spiel.

Johann Christoph Friedrich von Schiller (1759 - 1805)

Da im Zuge der technischen Weiterentwicklung klassische mobile Systeme, wie PDAs oder Mobiltelefone, zum einen immer leistungsfähiger werden und daher mehr Funktionen stationärer Desktop PCs übernehmen können und zum anderen der Preis für diese Technik fällt, erfahren solche mobilen Systeme eine steigende Verbreitung [Mat01]. Daher stellt gerade diese neue Generation mobiler Geräte ein interessantes Zielsystem für Spielentwickler dar, da sich diese Geräte im Gegensatz zu Notebooks jederzeit aufgrund ihres geringen Gewichts und den kleinen Abmaßen beim Nutzer befinden. Durch die Verbindung dieser Systeme mit Sensoren zur Positionsbestimmung wie GPS entstand ein vollkommen neues Spielgenre, die sogenannten „Pervasive Games". Dabei steuert der Spieler seinen Avatar nicht mehr länger über ein Steuergerät, sondern indem er sich selbst durch die reale Welt bewegt. Neue Entwicklungen bei diesen Spielen versuchen eine größere Plattformunabhängigkeit zu erreichen, indem das Spiel vollständig in einem Webbrowser des mobilen Systems abläuft und alle Daten an einen zentralen Server weitergeleitet werden. Dadurch werden mobile ortsbasierte Browserspiele möglich.

In der vorliegenden Diplomarbeit wird der Stand der Forschung im Bereich mobiler ortsbasierter Spiele analysiert und die bestehenden Konzepte erfasst und kategorisiert. Dabei wird bei den bestehenden Konzepten besonders auf die Anforderungen hinsichtlich der Team-Aspekte - z.B. Teilziele, die gleichzeitig von verschiedenen Mitspielern in Angriff genommen werden müssen - der temporalen Aspekte - z.B. eine Deadline bis zu der eine Aufgabe erfüllt werden sein muss - und der Möglichkeiten, online mit räumlich entfernten Mitspielern zu interagieren - z.B. Kommunikationsmöglichkeiten im Spiel - geachtet. Da sich nicht jedes Konzept für den praktischen Einsatz im Webbrowser eignet, und sich manche Konzepte sogar überhaupt nicht umsetzen lassen, da die benötigten Ressourcen weit über die eines Smartphones hinausgehen oder zusätzliche Hardware benötigt wird, werden die gefundenen Spiele in Hinblick auf ihre Realisierbarkeit bewertet.

Basierend auf diese Analyse wird ein Framework implementiert, das die Entwicklung mobiler ortsbasierter Browserspiele unterstützt und die zentralen Aspekte der Konzepte, die sich für den Einsatz im Browser eignen, als Bausteine zur Verfügung stellt. Aufbauend auf dieses Framework wird beispielhaft ein Prototyp eines mobilen ortsbasierten Browserspiels entwickelt. Abschließend wird eine Evaluation dieses Spiels durchgeführt und hinsichtlich der aufgestellten Anforderungen an mobile ortsbasierte Spiele kritisch bewertet.

Als technische Umgebung wird ein Nokia Internet Tablet verwendet, dessen Webbrowser bereits um GPS-Funktionalität erweitert wurde. Dieses Gerät wird in Abschnitt 1.1.2 vorgestellt.

1.1. Hintergrund

1.1.1. Das Spiel

Zunächst sollte an dieser Stelle geklärt werden, was man unter dem Begriff „Spiel“ versteht und welche Rolle Spiele im Leben der Menschen einnehmen. Danach können mobile ortsbasierte Browserspiele kurz beschrieben und in diesen Rahmen eingeordnet werde.

Was ist ein Spiel?

[Mey07] definiert ein Spiel als „Tätigkeit, die ohne bewussten Zweck lediglich aus Freude an ihr selbst ausgeübt wird und mit Lustempfindungen verbunden ist“. Auf Grund dieser sehr weit gefassten Definition des Begriffs, sieht die Literatur auch in kulturellen Handlungen, wie dem Schau**spiel**, in religiösen Ritualen, wie dem Feiern von sakralen Festen oder in Kunstformen, wie der Poesie, eine Form des Spiels [Hui56].

Diese Arbeit beschränkt sich allerdings auf Spieltypen, die dem allgemeinen Verständnis von Spielen entsprechen und sich in den Reihen der Geschicklichkeits-, Kraft-, Verstands- und Glücksspiele finden lassen. Darstellungs- und Aufführungsspiele werden nur dann näher betrachtet, wenn sie einen Nebenaspekt einer der genannten Gruppen darstellen. Obwohl Tätigkeiten mit Trainings- oder Lerncharakter der obigen Definition nicht entsprechen, da diese nicht „ohne bewussten Zweck [...] ausgeübt“ [Mey07] werden, werden diese dennoch als Spiele angesehen. Huizinga definiert ein Spiel allerdings nicht nur über dessen Zweck, sondern dadurch, dass es sich dabei um „eine freiwillige Handlung oder Beschäftigung, die [...] freiwillig angenommenen [wird], aber unbedingt bindenden Regeln“ [Hui56, S. 34] folgt, handelt.

Warum spielen wir?

Auch wenn man allgemein sagen kann, dass das Spiel in der Natur aller Lebewesen liegt, um einen Trieb zu befriedigen [Hui56], kann man noch weitere Funktion von Spielen finden. Anhand dieser Funktion lassen sie sich in eine oder mehrere der folgenden fünf Kategorien einteilen:

Bewegungsspiele können erstrecken sich vom einfachen „Herumtollen“ bis hin zu anspruchsvollen Klettertouren. Diese Art des Spiels dient der Verbesserung der Motorik und Erprobung der Körperkraft.

Phantasie- und Nachahmungsspiele werden in erster Linie von Kindern ausgeübt, indem sie Berufe oder Verhaltensweisen von Erwachsenen spielerisch nacheifern. Während Phantasiespiele sich mit erdachten Situationen beschäftigen, werden bei Nachahmungsspiele wirklich erlebte Situation möglichst exakt reproduziert. Durch diese Imitation wird eine Vielzahl verschieden Verhaltensweisen unbewusst erlernt und eingeübt.

Soziale Spiele finden in einer meist unveränderlichen, homogenen Gruppe statt, da diese Spiel zum einen das Gefüge der Gruppe festigen und zum anderen inhaltlich durch die Interessen der Gruppe bestimmt werden. Ziel dieser Spiele ist es durch gespieltes Teilen und Wetteifern Reibungspunkte innerhalb der Gruppe zu reduzieren.

Wettkämpfe finden sich in den unterschiedlichsten Arten von Spielen, sobald mehr als ein Spieler involviert ist, da der Wunsch besser als der andere zu sein und diesen in allem zu übertreffen, schon von klein auf gefördert wird und so die Position in der Gesellschaft gefunden werden kann.

Auch wenn jegliche Art des Spiels letztendlich scheinbar nur zum Abbau der Monotonie genutzt wird, sieht man leicht, dass Spiele deutlich mehr darstellen und sich eine intensive Auseinandersetzung mit diesen lohnt. [Mil73]

Auch wenn sich zwar einige Beispiele von computergestützten Spielen aus allen diesen Kategorien (hauptsächlich auf dem Konsolenmarkt) finden lassen - als Beispiele seien an dieser Stelle das Bewegungsspiel „Wii Fit“ von Nintendo [Nino8], bei dem mittels BMI für den Spieler ein individuelles Trainingsprogramm erstellt und die Ausführung dieser Übungen bewertet wird, und das Nachahmungsspiel „Rock Band“ von Harmonix Music Systems [Haro7], bei dem der Spieler in die Rolle eines Musikers schlüpft und diesen möglichst exakt imitieren muss, erwähnt - aber diese Arbeit zielt in erster Linie auf Spiele mit kompetitivem Charakter - sei es gegen reale Kontrahenten oder CPU-gesteuerte Gegenspieler - ab. Wobei selbstverständlich durch die Natur der mobilen ortsbasierten Spiele immer eine Bewegungskomponente enthalten ist.

Pervasive Games

Unter „Pervasive Computing“ versteht man ein System, das dem Nutzer einen Zugriff auf die gewünschten Informationen zu jeder Zeit und an jedem Ort gestattet. Diese Systeme müssen nicht nur mobil, sondern auch handlich genug sein, damit der Nutzer sie auch immer und überall zur Hand hat. Des Weiteren ist eine drahtlose Internetverbindung zwingend erforderlich, um das Gerät mit zusätzlichen, räumlich entfernten (virtuellen) Informationen zu versorgen und diese in physische Gegenstände der realen Welt umzuwandeln. [BBG+00]

„Pervasive Games“ gehen dabei meist einen Schritt weiter, indem sie nicht nur Informationen aus der virtuellen Welt abrufen und materialisieren, sondern auch Informationen über den Spieler in die virtuelle Welt einbauen, um so die Grenze zwischen der „Spielwelt“ und der Realität möglichst fließend zu gestalten. So können beispielsweise virtuelle Schätze in der Umgebung des Spielers erzeugt werden, die dieser Einsammeln kann, in dem er sich in der realen Welt dorthin bewegt, wie man in Abbildung 1.1 sehen kann.

Abbildung 1.1.: Human Pacman (Quelle: vgl. [Mixo8])

Es gibt allerdings mehrere Ansätze, ein solches Spiel zu entwerfen. Eine Möglichkeit besteht darin, bereits bestehende „normale“ Spiele so zu erweitern, dass sie mobil und ortsbasiert ablaufen. Andere

Ansätze setzen auf Interaktionen mit den Mitspielern, die man auf seiner Reise durch die virtuelle Welt trifft und sich mit diesen in der realen Welt unterhalten kann. Schließlich kann die Technik auch zur reinen Wissensvermittlung verwendet werden, indem abhängig von der Position des Spielers Informationen über seine Umgebung auf dem Display des Spielgeräts eingeblendet werden. Bei der Gestaltung des Informationsaustauschs zwischen den beiden Welten sind dabei der Kreativität der Entwickler keine Grenzen gesetzt und daher sind auch viele weitere Formen der Pervasive Games vorstellbar. [BML05]

1.1.2. Die Nokia Internet Tablet Produktreihe

Mobile Geräte sind aus dem heutigen Alltag vieler Menschen nicht mehr wegzudenken. Dabei übernehmen sie immer öfter Aufgaben, die über die reine Kommunikation hinausgehen, so dass diese immer weiter in den Hintergrund rückt. Neben Funktionen, wie eines elektronischen Adressbuches oder der Wiedergabe von Filmdateien, ist in erster Linie die Möglichkeit jederzeit und überall Informationen aus dem Internet zu beziehen von Interesse. [SHA+06] Nokias Antwort auf diese neue Generation von mobilen Geräten die der „always on“ und „always online“ Philosophie folgen, stellt die Internet Tablet Produktreihe dar. Die Nokia Internet Tablet Produktreihe zeichnet sich durch einen breiten Bildschirm und eine geringe Größe aus. In Verbindung mit der drahtlosen Konnektivität über WLAN oder Bluetooth eignen sich diese Geräte daher ausgesprochen gut zur mobilen Internetkommunikation.

Abbildung 1.2.: Das Nokia N810 (Quelle: [Zif07])

Für diese Arbeit kommt ein Nokia N810 (Abbildung 1.2) zum Einsatz. Dieses Gerät ermöglicht eine dauerhafte Internetverbindung - ein WLAN-Zugang oder ein internetfähiges Handy mit Bluetooth-Funktionalität vorausgesetzt. Die Einbindung der Position des Nutzers kann mittels GPS-Signals erfolgen. Das N810 verfügt bereits über einen internen GPS-Empfänger. Zusätzlich kann optional ein stärkeres, externes GPS-Empfangsgerät mittels Bluetooth angebunden werden, um Verbindungsprobleme zu minimieren. [Nok08]

1.2. Aufbau dieser Arbeit

Diese Arbeit ist folgendermaßen strukturiert: Das zweite Kapitel soll einen Überblick über vergleichbare Studien liefern. Hier werden die möglichen Konzepte der Entwicklung von mobilen ortsbasierten Spielen und Browserspielen vorgestellt und einige der bereits bestehenden Titel im Detail betrachtet. Dabei wird bereits auf die Verwendungsmöglichkeiten in einem Browser Rücksicht genommen. In Kapitel drei wird die Technik, die zum Einsatz kommen soll, vorgestellt. Neben den Komponenten für eine drahtlose Vernetzung (z.B. WLAN oder Bluetooth) und für die Positionsbestimmung, wird auch auf aktuelle Konzepte zur Webprogrammierung in Form von AJAX in Verbindung mit dem Google Web Toolkit Framework oder Adobe Flash eingegangen. Um die Positionsdaten verwenden zu können, wird ebenfalls die DCCI-Spezifikation für den Datenaustausch benötigt. Kapitel vier arbeitet die benötigten Anforderungen dieser Spiele heraus. Diese werden in verschieden Kategorien eingeteilt und in einem Forderungskatalog zusammengefasst. Auf das entwickelte Framework wird in Kapitel fünf näher eingegangen, bevor in Kapitel sechs ein konkreter Prototyp eines mobilen ortsbasierten Browserspieles vorgestellt wird, der mit Hilfe des Frameworks implementiert wurde. Kapitel sieben vergleicht das erstellte Spiel mit dem in Kapitel drei aufgestellten Anforderungskatalog und setzt sich mit dem Ergebnis kritisch auseinander, ehe Kapitel acht mit einem Fazit diese Diplomarbeit abschließt.

1.3. Danksagung

An dieser Stelle möchte ich folgenden Personen (in alphabetischer Reihenfolge) danken. Ohne sie wäre meine Arbeit in dieser Form nicht möglich gewesen. Vielen Dank!

- Dem Community Manager **„Agge“** von Travian, für die Informationen zu den eingesetzten Programmierungstechniken bei seinem Browserspiel,
- **Dipl.-Inf. Andreas Brodt**, für seine adäquate Unterstützung bei meiner Arbeit,
- **Dipl.-Inf. Fabian Kaiser**, der mich auf dieses Thema aufmerksam gemacht hat,
- **Fabian Loschek** von Comunio.de, da er mich mit Informationen zum technischen Aufbau seiner Plattform versorgte,
- **Prof. Dr. Mitschang**, für die Zulassung dieses Themas zur Diplomarbeit,
- meiner Schwester, **Bettina Stach**, dafür dass sie diese Arbeit Korrektur gelesen hat,
- meinen Eltern, für ihre Unterstützung bei meiner Diplomarbeit und ihr Verständnis für meine Launen während dieser Zeit und schließlich
- meinen Freunden, die immer akzeptiert haben, wenn ich mich arbeitsbedingt häufig rar machen musste

Kapitel 2

Verwandte Arbeiten

Diese Arbeit steht mit zwei unterschiedlichen Themengebieten im Zusammenhang. Zum einen beschäftigt sie sich mit mobilen ortsbasierten Anwendungen im Allgemeinen und Spielen in Speziellen, wie sie bereits in Abschnitt 1.1.1 vorgestellt worden sind, und zum anderen mit den unterschiedlichen Formen „herkömmlicher" Browserspiele. Daher sollen an dieser Stelle die Ergebnisse bereits bestehender Arbeiten aus beiden Themenkomplexen vorgestellt werden. Zunächst sollte jedoch genauer betrachtet werden, was ein Computerspiel exakt auszeichnet.

2.1. Computerspiele

Bevor mit dem Design und der Implementierung eines Spiels begonnen werden kann, muss zunächst ein Konzept erstellt werden, in welche Richtung sich das endgültige Produkt entwickeln soll. Dazu müssen Entscheidungen getroffen werden, wie beispielsweise welches Spielprinzip verfolgt werden soll, oder welches Genre man bedienen möchte. Diese Überlegungen können aber nicht unabhängig voneinander getroffen werden, da sich einige dieser Charakteristiken gegenseitig ausschließen oder einander voraussetzen. Des Weitern darf man nie aus dem Auge verlieren, dass man für eine angestrebte Zielgruppe produziert und daher das Spiel auch auf diese zurechtschneiden muss. Nicht jeder Kunde interessiert sich für jedes Spiel gleichermaßen. [Bato2]

Daher werden im Folgenden zunächst die möglichen Ausprägungen verschiedener, wichtiger Charakteristiken, wie sie in bestehenden Spiele vorkommen, betrachtet, bevor auf die speziellen Merkmale bei Pervasive Games und Browserspielen eingegangen wird. Von diesen Faktoren hängen auch die möglichen Anforderungen an ein Framework entscheidend ab.

2.1.1. Charakteristiken von Computerspielen

[Napo6] charakterisiert Spiele anhand folgender Kriterien, die sich sowohl auf normale, als auch auf Pervasive Games anwenden lassen:

Das Spielprinzip wird anhand mehrerer Spielregeln definiert. Dabei gibt ein Regelwerk in jeder Situation vor, welche Aktionen der Spieler ausführen darf und welche Konsequenzen sein Handeln für ihn hat.

Die Spielerzahl hat einen maßgeblichen Einfluss auf das Spielsystem. So lassen sich Spiele zunächst in Einzel- und Mehrspielerspiele einteilen. Als Einzelspielerspiel gelten neben „echten“ Einzelspielerspielen, also Spiele, mit nur einer agierenden Person, auch solche, mit nur einem einzigen menschlichen Spieler. Bei den letztgenannten übernimmt dabei das Programm die Kontrahenten, die mit dem Spieler in Aktion treten. Auch Mehrspielerspiele lassen sich in zwei Kategorien einteilen - rundenbasierte Spiele, bei denen die einzelnen Spieler abwechselnd agieren, und solche, bei denen alle Spieler gleichzeitig ihre Aktionen tätigen dürfen.

Das Setting gibt den inhaltlichen Rahmen, in dem das Spiel eingebettet ist vor. Da das Setting keinen direkten Einfluss auf den Spielablauf hat, muss dieses von einem Framework nicht weiter beachtet werden, obwohl es für den Erfolg eines Spiels von entscheidender Bedeutung sein kann.

Das Genre beschreibt um was für eine Art Spiel es sich handelt. Unter einem Genre werden viele Produkte zusammengefasst, die Gemeinsamkeiten hinsichtlich Spielprinzip, Spielerzahl, Setting, Grafik, Bedienung, usw. aufweisen. Häufig lassen sich Spiele allerdings nicht eindeutig einer Kategorie zuweisen. So entstehen (neue) Genre-Mixe, wie beispielsweise dem Action-Adventure, das neben Action wie schnellen Lauf-, Spring- und Schusssequenzen auch Adventureelementen und Rätseleinlagen beinhaltet. Häufig kommt eine Einteilung in diese Oberkategorien zum Einsatz:

- Unter **Actionspielen** werden diejenigen Spiele zusammengefasst, die in erster Linie auf Reaktionsschnelligkeit und Geschicklichkeit setzen.
- Bei **Abenteuerspielen** handelt es sich um Spiele, bei denen der Spieler eine fesselnde Geschichte durchlebt und dabei regelmäßig Rätsel lösen muss, um diese voranzutreiben.
- **Strategiespiele** fordern den Spieler dadurch, dass er in einer virtuellen Welt mit mehreren Wettbewerbern um unterschiedliche Ressourcen konkurriert.
- **Simulationen** versetzen den Spieler in die Lage (möglichst) realitätsnah eine bestimmte Tätigkeit auszuüben, die er im wirklichen Leben nicht oder nur sehr schwer praktizieren kann.
- Unter **sonstige Spiele** fallen alle Programme, die bislang in kein Genre passten. Da diese Spiele allerdings relativ selten sind, existiert für sie kein eigenes Genre.

Diese Genres sind allerdings ausgesprochen dynamisch und können um neue Genres erweitert werden, wenn sich eine neue Gattung auf dem Markt durchsetzt, oder verkürzt werden, sollte eine dieser Genres „aussterben“.

Die Grafik eines Spiels muss anhand mehrerer Faktoren, wie den verfügbaren Projektressourcen, der Erwartungen der Kunden abhängig vom Genre oder der Leistungsfähigkeit des Zielsystems, ab. Damit ist allerdings nicht die Qualität oder der Stiel der Grafik gemeint, sondern die Art der Darstellung. Dabei unterscheidet man in der Regel zwischen den folgenden drei Präsentationsarten:

- **Textbasierte Darstellung**, bei der der Spieler seine Spielfigur bewegt, indem er eindeutige, im Vorfeld definierte, Kommandos eingibt. Die Reaktion auf diese Handlung wird in der Regel durch eine Textausgabe dargestellt.
- **2D-Darstellung**, bei der die Spielhandlung mittels zweidimensionalen Abbildungen abläuft.
- **3D-Darstellung**, bei der die komplette virtuelle Welt dreidimensional modelliert wurde.

Eine Spielunterbrechung kann neben einer kurzen Pause, in der das Spiel ohne Aktionen seitens des Nutzers auf dem System weiterläuft, auch eine beliebig lange Unterbrechung, während der sogar das komplette System ausgeschaltet werden kann, verstanden werden. Daher hängt mit der Unterbrechung auch das Speichersystem, also die Möglichkeit den momentanen Speicherinhalt mit allen Veränderungen der virtuellen Welt zu sichern und später wiederherzustellen, zusammen. Dabei können mehrere Speichersysteme zum Einsatz kommen, wie sie von [Bro04] vorgestellt werden:

- **Keine Speicherung**: Wie der Name schon andeutet, wird keinerlei Speicherung zugelassen. Gerade in Mehrspielerspielen trifft man dieses System an, da so der Spielfluss am Laufen gehalten wird. Um dem Spieler dennoch ein die Möglichkeit zu geben, seine Leistungen persistent zu halten, kann sein letzter Punktestand in einer Datenbank festgehalten werden.
- **Fortschrittsspeicherung**: Nur an bestimmten Punkten im Spiel ist eine Speicherung gestattet. Dabei wird nicht die komplette Welt gespeichert, sonder nur der Fortschritt des betreffenden Spielers
- **Komplettspeicherung**: Der Zustand jedes Objekts der virtuellen Welt wird gespeichert. Dadurch befindet sich der Spieler nachdem er einen alten Spielstand geladen hat, in exakt der gleichen Situation, wie sie zum Zeitpunkt der Speicherung war.

2.1.2. Spezielle Charakteristiken von Pervasive Games

Neben den in Abschnitt 2.1.1 behandelten Charakteristiken betrachtet [Nap06] auch spezielle, für Pervasive Games entscheidende Faktoren, die sich zum Teil aus der zum Einsatz kommenden Hardware ergeben:

Die Mobilität ist für diese Spiele entscheidend, da, wie in Abschnitt 2.2.1 erläutert, es für pervasive Anwendungen jeder Art unverzichtbar ist, dass diese ortsbasiert und mobil sind. Daher muss ein Spielgerät sich ohne größere Probleme durch den realen Teil der Spielwelt transportieren lassen. Für jede Art der Anwendung muss eine passende Hardware gefunden werden. Diese kann von Notebooks - für rechenintensive Programme - bis hin zu Geräten ohne Display reichen, die für ein einfaches Suchspiel eingesetzt werden können, bei dem durch Lichtsignale anzeigt wird, ob man sich seinem Ziel nähert oder davon entfernt.

Die Kommunikation kann auf Grund der Mobilität des Spielers nur über drahtlose Systeme, wie WLAN oder Bluetooth, erfolgen. Unter „Kommunikation“ fällt im Verständnis dieser Arbeit aber nicht nur Gespräche mit eventuellen Mitspielern, sondern ebenfalls der Datenaustausch mit einem entfernten Spielserver, den es gerade bei Browserspielen zwangsläufig geben muss. Dabei sollte die Konfiguration möglichst im Hintergrund ablaufen, um einen technisch unversierten Nutzer nicht unnötig zu überfordern. Dies ist gerade dann entscheidend, wenn mehrere Techniken, je nach Verfügbarkeit, zum Einsatz kommen sollen. Bereits bei dem Systementwurf sollte daran gedacht werden, dass diese Kommunikationskanäle gestört sein können oder sogar ganz ausfallen. In diesem Fall wäre es wünschenswert dem Spieler dennoch zumindest ein eingeschränktes Weiterspielen zu ermöglichen.

Die Weltmodellierung ist ebenso wie die Mobilität ein für Pervasive Games unverzichtbarer Faktor. Die Anreicherung der realen Welt um virtuelle Inhalte kann nur dann erfolgreich sein, wenn das System jederzeit erkennen kann, wo sich der Spieler momentan aufhält und seine Position mit der der erschaffenen Gegenständen abgleicht. Dabei können zwei Einsatzgebiete unterschieden werden:

- Wird versucht, die Umgebung des Spielers als Spielfeld zu verwenden, also ihm bekannte Straßen, Gebäude, usw. möglichst originalgetreu in die Spielwelt zu übernehmen, so spricht man von der **Modellierung eines realen Kontextes**. In diesem Fall reicht eine reine Übernahme der Positionsdaten - beziehungsweise deren Änderung - nicht aus. Sie müssen nach der Übermittlung automatisch um Informationen des realen Kontextes in dem der Nutzer sich im Moment befindet aufbereitet werden - „Hat ein bestimmtes Gebäude gerade geöffnet und kann betreten werden?“ wäre zum Beispiel eine solche wichtige Information - um so die momentan möglichen regelkonformen Aktionen zu ermitteln.
- Ist hingegen lediglich die Bewegungsrichtung entscheidend und das Spiel findet fern ab von dem realen Umfeld des Spielers statt, so heißt dies **Modellierung eines virtuellen Kontextes**. Diese virtuelle Welt kann für alle Spieler gleich aussehen, unabhängig davon, wo das Spiel betreten wird. Diese Modellierungsart ist zwar bedeutend einfacher, da keinerlei Kontextinformationen neben der Positionsänderung benötigt werden, aber dadurch auch fehleranfälliger. Beispielsweise können in der echten Welt unüberwindbare Hindernisse existieren, die in der virtuellen Welt nicht vorkommen. Außerdem birgt das System auch Gefahren für den Spieler, da dieser sich auf seiner Tour durch die reale Welt ausschließlich auf das Display seiner Spielkonsole konzentrieren muss.

Wie bereits in Abschnitt 2.3 vorgestellt wurde, gibt es abhängig von der Umgebung mehrere Möglichkeiten, die Position des Spielers zu ermitteln. Da die meisten mobilen ortsbasierten Spiele im Freien stattfinden und möglichst ohne großen Zusatzaufwand auskommen sollten, wird auf ein GPS-System zurückgegriffen. Aber auch bei der Positionsbestimmung kann es zu Ausfällen kommen, weshalb auch hier ein Notfallplan erstellt werden sollte.

Der Sensoreinsatz ist zwar, wie in Abschnitt 2.3.5 betrachtet, sehr reizvoll, da viele neue Spielformen und Eingabemöglichkeiten denkbar sind - man stelle sich den Einsatz einer Kamera zum „Abschuss“ eines Ziels oder ein Fitnesstraining, bei dem auch der Puls des Spielers gemessen und in den Kontext miteinbezogen wird, vor - findet aber nur in den wenigsten Systemen Verwendung. Dies liegt einerseits sicherlich daran, dass die automatische Analyse solcher Daten sehr schwierig sein kann. Aber auch die relativ geringe Verbreitung dieser zusätzlichen Sensoren in den Endgeräten macht den Einsatz wenig erfolgversprechend.

Weitere Technologien, wie das in Abbildung 2.1 dargestellte Head Mounted Display finden gerade im wissenschaftlichen Bereich Anwendung. Dadurch kann zwar eine noch stärkere Verbindung von der virtuellen Welt mit der realen geschaffen werden. Aber neben dem bereits bei dem Einsatz weiterer Sensoren besprochenen Problem der geringen Verbreitung, kommt hier des weiteren der sehr hohe Preis und die starke Einschränkung der Mobilität zu tragen. Ein kurzes Spiel zwischendurch ist praktisch unmöglich.

Browserspiele kommen ohne eine Installation aus, da sie in der Regel auf einem Server ablaufen. Dadurch kann sich das Spiel auch weiterentwickeln, wenn der Client es lokal beendet, um so sogar Endlosspiele zu ermöglichen. [Lubo8] Meist sind sie für eine sehr große Anzahl von Spielern, die gleichzeitig agieren können, ausgelegt. Bei der Umsetzung muss vollständig auf Plattformen der herkömmlichen Spielentwicklung wie DirectX, die eine kontinuierliche und schnelle Interaktion mit dem Spielern und Echtzeitgrafik erheblich erleichtern, verzichtet werden. Die vollständige Spiellogik, Grafik-Engine, usw. muss mit den gängigen Webtechnologien, wie sie in Abschnitt 3.3 vorgestellt werden, implementiert werden. [Wiko8e]

Bei dem Entwurf des mobilen ortsbasierten Browserspiels müssen also zunächst diese sechs Punkte beachtet werden und es muss entschieden werden, welche Richtung das Spiel einschlagen soll.

2.2. Begriffsabgrenzung

Bevor an dieser Stelle näher auf die unterschiedlichen, bestehenden, mobilen, ortsbasierten Anwendungen eingegangen werden kann, müssen einige Begrifflichkeiten, die in diesem Umfeld regelmäßig auftauchen, definiert werden. Danach sollen die technischen Entwicklungen in diesem Bereich präsentiert werden, ehe ein Überblick über aktuelle Spieleentwicklungen das Unterkapitel abschließt.

2.2.1. „Pervasive“ und „Ubiquitous“ Systeme

Im Umfeld der mobilen ortsbasierten Anwendungen finden sich meist die Begriffe des „Pervasive Computing“ und des „Ubiquitous Computing“, die häufig als Synonyme verwendet werden. Darunter werden Computersysteme zusammengefasst, die einen dauerhaften und omnipräsenten Zugriff auf die persönlichen Daten des Nutzers und die Anreicherung dieser mit zusätzlichen Informationen aus externen Quellen verstanden [Mat01] - genau wie es der Wortursprung bereits vermuten lässt. Während „pervasive“ in etwa so viel wie „überall vorhanden“ bedeutet, wird „ubiquitous“ mit „allgegenwärtig“ übersetzt. Dennoch unterscheiden sich diese Begriffe in entscheidender Weise.

Auch wenn das Verständnis von Pervasive Computing mitunter sehr vielfältig ist und von der Forderung nach Mobilität vormals stationärer Geräte bis hin zu Systeme, die ihren Funktionsumfang selbstständig an ihre Umgebung anpassen, reicht, so lassen sich drei Forderungen an diese Geräte ableiten:

1. Ein freier Zugriff auf Informationen aus externen Datenlagern.
2. Die Software soll auf die Bedürfnisse des Nutzers und nicht auf die Fähigkeiten der vorhandenen Hardware ausgerichtet sein.
3. Die Hauptaufgabe des Systems besteht in der Informationsversorgung.

Dies kann nur dann gewährleistet werden, wenn die Systeme sowohl mobil, als auch ortsbasiert sind. [BBG+00]

Die Forderungen an ubiquitäre Anwendungen gehen allerdings weit über eine allgegenwärtige und dauerhafte Verfügbarkeit von Daten hinaus. Mark Weiser, der den Begriff des „Ubiquitous Computing“ prägte, stellte sich das perfekte System nahezu unsichtbar vor, obgleich es überall verfügbar ist [Wei93]. Daraus ergib sich, dass Ubiquitous Systeme einfach und möglichst „natürlich“ bedient werden können, deren Funktionsweise automatisch der Umgebung des Nutzers angepasst wird und diese wichtige Informationen selbstständig beziehen und aufbereiten können. Weitere Forderungen an die Ausmaße der Systeme, damit diese überall verfügbar sein können und an die Zuverlässigkeit, damit auch die dauerhafte Nutzbarkeit gewährleistet ist, sind zusätzlich vorstellbar. [AM00] Im Folgenden wird das Umfeld der Ubiquitous Computing nicht weiter betrachtet und ausschließlich von Pervasive Games gesprochen.

2.2.2. „Virtual“ und „Augmented“ Reality

Betrachtet man die Vorstellung der mobilen ortsbasierten Spiele in Abschnitt 1.1.1, so fühlt man sich unweigerlich an Virtual Reality Anwendungen erinnert, bei denen der Computer eine virtuelle Welt schafft, die vom Spieler betreten wird und in der er agieren kann. In einer solchen virtuellen Umgebung kann der Nutzer jedoch nur mit Dingen interagieren, wenn das System auch eine Aktion erwartet. Dies ergibt sich aus der Tatsache, dass alle Gegenstände in der virtuellen Welt nicht existieren, sondern nur vom Computer erzeugt werden. Die „echte“ Welt wird nach Möglichkeit vollständig

ausgeblendet. Daher muss bei der Entwicklung einer VR Anwendung neben dem Rendering und den Eingabemöglichkeiten besonders auf eine realistische Wiedergabe des Verhaltens aller implementierter Objekte gelegt werden. Soll der Nutzer vollkommen in der erschaffenen Welt versinken, sind natürlich andere Ein- und Ausgabegeräte als bei normalen Anwendungen von Nöten. [SRH05] Ein solches Gerät stellen die Head Mounted Displays dar, wie sie in Abbildung 2.1 zu sehen sind. Dabei wird der sichtbare Ausschnitt der virtuellen Welt auf ein Display direkt vor den Augen des Nutzers projiziert. Kopfbewegungen können von Sensoren im Helm wahrgenommen werden und entsprechend in der erschaffenen Umgebung umgesetzt werden. [PSY97]

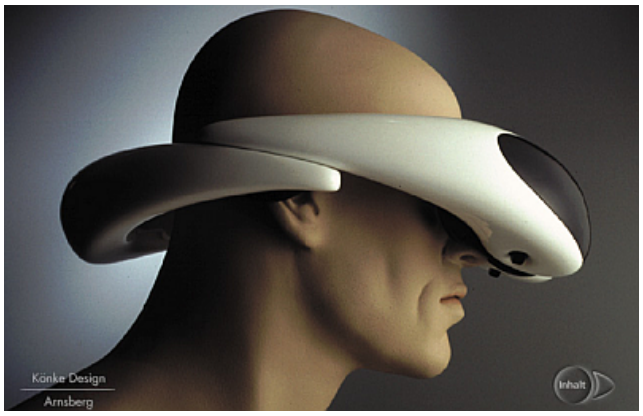

Abbildung 2.1.: Design eines Head Mounted Displays (Quelle: vgl. [Pro07])

Jedoch zielt diese Arbeit nicht auf die oben genannten Techniken ab. Vielmehr sollen hier Anwendungen aus dem Bereich der „Augmented Realities“ , also eine um virtuelle Objekte angereicherte Realität, betrachtet werden. Worin der Unterschied genau liegt, soll im Folgenden kurz eingegangen werden.

Da Augmented Reality Anwendungen lediglich, wie es der Name schon sagt, die Welt des Nutzers erweitern und nicht einen eigene, vollständig virtuelle Umgebung erschaffen wollen, eignen sich die oben genannten Ein- und Ausgabegeräte, die versuchen die Realität auszublenden, nicht. Als eine geeignete Bedienung könnte beispielsweise ein durchsichtiges HMD verwendet werden, das nur für die Darstellung der virtuellen Objekte zuständig ist. Die Eingabe darf den Anwender nicht in seinen natürlichen Bewegungen einschränken - schließlich soll er sich ja weiterhin in seiner kompletten natürlichen Umgebung bewegen können - und auch eine Interaktion mit den virtuellen Objekten sollte sich nicht von der mit den real existierenden unterscheiden. Bedingt durch die Eigenschaft, dass sich die virtuelle Umgebung nahtlos in die Realität einfügt, laufen solche Anwendungen zwangsweise in Echtzeit und in dem Umfeld in dem der Nutzer sich befindet ab. Man könnte also sagen, dass die Augmented Reality eine Spezialform einer Virtual Reality darstellt. [Azu95]

Lokale ortsbasierte Anwendungen finden daher immer in einer Augmented Reality statt, wobei das Realitätsempfinden sehr stark von der Einbindung der künstlich erzeugen Objekte und der verwendeten Hardware abhängt - wie man dies auch im Browser und ohne spezielle Ein- und Ausgabegeräte (zumindest zum Teil) erreichen kann, wird in Kapitel 6 vorgestellt.

2.3. Mobile ortsbasierte Anwendungen

Wie man bei der Definition von Pervasive Computing in Abschnitt 2.2.1 sehen konnte, ist es für diese Systeme - und damit auch für die in dieser Arbeit behandelten Pervasive Games - unverzichtbar, dass die verwendete Hardware ihr augenblickliches Umfeld erkennen und darauf reagieren kann. Auch wenn es wünschenswert wäre, dabei nicht nur die Ortsdaten zu berücksichtigen, ist es sehr schwer weitere Aspekte, wie beispielsweise die augenblicklichen Wünsche und Bedürfnisse des Nutzers, zu messen. [Kaa03] Der Grundstein für mobile ortsbasierte Anwendungen wurde bereits 1955 gelegt, als Edward O. Thorp das erste tragbare Computersystem zur Analyse von Roulettespielen entwickelte, auch wenn dabei freilich noch von keinem Pervasive System gesprochen werden konnte [Tho98]. Im Folgenden soll ein kurzer Überblick über einige ortsbasierte Anwendungen gegeben werden.

2.3.1. Active Badge Location System

Die bereits seit 1989 von dem Olivetti Research Laboratory in Cambridge entwickelten Active Badges stellen einen ersten ernsthaften Versuche dar, wie eine ansatzweise ubiquitäres Verarbeitung realisiert werden kann. Dabei wird nicht versucht den Nutzer selbst zu orten, sondern einen Sender, der immer mitgeführt werden muss. [BZ98] Dieser Sender schickt in einem bestimmten Zeitintervall ein eindeutiges Infrarot-Signal mit dessen Hilfe er identifiziert werden kann. Die Einsatzmöglichkeiten sind hierbei mannigfaltig und es können neben einer einfachen Zugangskontrolle oder der Erfassung aller anwesenden Personen auch Dienste, die das Wissen über den Aufenthaltsort einer Person ausnutzen, implementiert werden. Ein mögliches Einsatzszenario hierfür wäre eine Weiterleitung aller eingehender Anrufe nicht an den Anschluss der betreffenden Person, sondern an den Apparat, der ihr - bzw. viel mehr ihrem Sender - am nächstgelegenen ist. [WHFG92] Ein solcher Ausweis ist in Abbildung 2.2 zu sehen.

Abbildung 2.2.: Active Badge (Quelle: vgl. [Com02])

Dieses System kann allerdings nur innerhalb kleiner abgegrenzter Bereiche funktionieren, da ein Netzwerk aus Sensoren, die die Signale der Sender auswerten können, benötigt wird. Für Spiele eignet sich das System ebenfalls nur bedingt, da dem Nutzer keine Eingabemöglichkeiten gegeben werden und die Ausweise nur die ID senden, aber keine Daten empfangen können. Mit weiteren Hardwarekomponenten könnte dieses Problem allerdings umgangen werden.

2.3.2. ParcTab

Während sich die Active Badges noch sehr von der Hardware, die im Folgenden als Grundlage für das Pervasive Game dienen soll, ähnelt das 1993 von Xerox PARC entwickelte System heutigen PDAs sehr stark. Die als ParcTabs bezeichneten Geräte kommunizieren, wie auch die Active Badges mittels Infrarot-Signale mit einem Netzwerk. Allerdings können sie auch aktiv bedient werden. Neben drei Knöpfen verfügen sie über ein druckempfindliches Display, das natürlich sowohl als Ein-, als auch als Ausgabegerät benutzt werden kann. Somit dient ein ParcTab nicht nur der Ortung der Mitarbeiter - und allen damit verbundenen bereits in Abschnitt 2.3.1 angesprochen Möglichkeiten -, sondern es können auch Anwendungen damit ausgeführt werden. Unter den implementierten Anwendungen befanden sich z.B. ein Kalender, ein Wörterbuch oder einem Programm zum Abrufen der aktuellen Wetterdaten. Dabei laufen die meisten Anwendungen allerdings nicht auf dem System selbst, sondern sie werden im Netzwerk ausgeführt und nur die Ergebnisse auf den ParcTab übertragen. [SAG+93] Die Bedienungsoberfläche ist in Abbildung 2.3 zu sehen.

Abbildung 2.3.: ParcTab Computer (Quelle: [SW95])

Scheinbar eignen sich diese System bereits für sehr einfache Spiele, jedoch funktioniert der ParcTab ebenfalls nur innerhalb kleiner abgegrenzter Bereiche, da er auf ein Netzwerk aus Infrarot-Sensoren angewiesen ist. Ohne diese würde nicht nur die Ortung ausfallen, sondern es könnten auch keine Anwendungen mehr ausgeführt werden, da die Rechenleistung des ParcTabs nicht ausreicht, um diese selbstständig ausführen zu können. Ein weiteres Problem stellt das sehr kleine Display dar, das mit 128 x 64 Pixel kaum in der Lage ist, größere Informationsmengen sinnvoll darzustellen.

2.3.3. GUIDE

Das GUIDE System soll an dieser Stelle als Vertreter für die vielen elektronischen Reiseführer, wie z.B. dem Cyberguide System, das an dem Georgia Institute of Technology entwickelt wurde (siehe [AAH+97]), vorgestellt werden. Das an der Lancaster University 1999 implementierte System wurde ausgewählt, da dieses nicht nur die Positionsdaten des Nutzers auswertet, sondern ebenfalls auf seine Vorlieben eingehen kann. Es soll vier Ansprüchen genügen:

1. Das System muss flexibel sein und darf dem Nutzer nicht nur vorgefertigte Führungen anbieten, sondern soll sich ihm anpassen.

2. Das System muss auf sein Umfeld reagieren können. Neben ortsbezogenen Daten, die dafür sorgen, dass der Nutzer die richtige Information zur richtigen Zeit erhält, werden auch die persönlichen Daten des aktuellen Nutzers ausgewertet, um die Informationen auf seine Interessen, sein Bildungsstand, sein Alter, usw. anzupassen.
3. Das System muss dynamisch auf unerwartete Situationen - z.B. veränderte Öffnungszeiten von Museen - reagieren können und den Nutzer darüber informieren.
4. Das System muss Interaktionsmöglichkeiten bieten, falls der Nutzer Rückfragen hat, die über die Leistungen von GUIDE hinausgehen.

Entsprechend dieser Punkte sehen die Anforderungen an die Hardware aus. Da es flexibel auf spontane Entscheidungen des Nutzers reagieren soll und keine vorgefertigten Standardinformationen präsentieren kann, ist die benötigte Datenmenge gewaltig. Aus diesem Grund entschied man sich dazu, keine Daten auf dem Gerät zu speichern, sondern zu jeder Zeit von einem zentralen Server zu beziehen. Daraus ergibt sich, dass die Hardware eine dauerhafte Verbindung zu dem Server haben muss. Diese kabellose Verbindung wird mittels WaveLAN Zellen realisiert. Auf Grund der relativ kleinen Zellen, kann auch die Ortung ausreichend genau über die Einwahlposition erfolgen. Hierfür werden also keine weiteren Komponenten benötigt. Für eine Interaktion mit dem Nutzer sollte allerdings noch eine Eingabemöglichkeit existieren. Unter Berücksichtigung dieser Anforderungen fiel die Wahl auf das Fujitsu TeamPad 7600 auf dem GUIDE in einem Webbrowser ablaufen kann. [CDM+00] Die recht intuitive Bedienung ist in Abbildung 2.4 abgebildet.

Abbildung 2.4.: GUIDE (Quelle: [Dis00])

Dank des guten Displays, der leistungsstarken CPU, der verwendeten Client-Server-Architektur mit den bereits vorgesehenen Möglichkeiten, um auf Ereignisse reagieren zu können und der problemlosen Erlernbarkeit der Bedienung [CMD02], eignet sich dieses System bereits recht gut für einfache mobile ortsbasierte Browserspiel. Lediglich die Positionsermittlung, die deutlich ungenauer ist als beispielsweise GPS, könnte sich zu einem Probleme entwickeln. Auch die Verbreitung von WaveLAN gerade in ländlichen Gebieten dürfte sehr beschränkt sein. Aber für Projekte wie REXplorer (siehe Abschnitt 2.4.10) sollte die verwendete Technik vollkommen ausreichen.

2.3.4. Navicore

Auch auf der für diese Arbeit herangezogenen Hardware wurden bereits pervasive Anwendungen implementiert. Eine davon ist das von der Firma Navicore Ltd. entwickelte Navicore System. Dabei handelt es sich um ein Navigationssystem, das die lokal gehaltenen Kartendaten um viele zusätzliche, aus dem Internet bezogene, Daten anreichert. Dadurch kann es trotz der vorliegenden, statischen Karten dynamisch auf Veränderungen, wie z.B. Staumeldungen reagieren. Zusätzlich können weitere kontextbezogene Daten, wie beispielsweise die nächste Raststädte, wenn der Nutzer Hunger bekommt, eingeblendet werden. Diese zusätzlichen Daten erhält das System direkt aus dem Internet, wenn das Internet Tablet eine Verbindung, sei es über Bluetooth oder WLAN, aufbauen kann. Die Positionsermittlung kann dank GPS-Empfänger direkt und sehr genau erfolgen. [Navo8b] Wie man in Abbildung 2.5 sehen kann, unterscheidet sich die grafische Anzeige nicht von der eines normalen Navigationsgeräts und auch der restliche Funktionsumfang - z.B. sowohl visuelle, als auch gesprochene Routenführung - entspricht dem Standard bei diesen Geräten.

Abbildung 2.5.: Navicore (Quelle: vgl. [Navo8a])

Dank des sehr großen, druckempfindlichen, Widescreen-Displays, der vorhandenen Tastatur, des integrierten GPS-Empfängers und der schnellen Internetverbindung, die durch WLAN auch eine weite Verfügbarkeit garantieren sollte, stellt das Nokia Internet Tablet eine sehr gute Plattform für mobile ortsbasierte Spiele dar.

2.3.5. „Technology for Enabling Awareness“

Wenn bislang von kontextbezogene Daten gesprochen wurde, so war dabei meist nur die Position des Nutzers gemeint. Jedoch könnte die Leistungsfähigkeit mobiler ortsbasierter Anwendungen erheblich gesteigert werden, wenn neben diesen auch weitere Daten berücksichtigt würden. [SF99] So wäre es wünschenswert, wenn eine Anwendung dem Nutzer bei Ankunft im Büro beispielsweise einen Überblick über seine Termine gibt, mittags das Essensangebot in der Kantine einblendet und abends seine möglichen Busverbindungen, inklusive aller aktueller Verspätungen, anzeigt. Dies ist aber nur möglich, wenn neben der Position - die sich in dem gegeben Szenario nicht einmal verändert haben muss - auch weitere Daten einfließen. Diese sollten, soweit möglich, vom System selbstständig erfasst werden können, ohne dass der Nutzer dies alles einstellen muss, wie es bei dem GUIDE Projekt aus Abschnitt 2.3.3) nötig war. Daher benötigt ein solches System einige weitere Komponenten. „Technology for Enabling Awareness“ (kurz TEA) wurde 2000 im Rahmen des ESPRIT

Projekts (siehe [Rog96]) u.a. von der Universität Karlsruhe fertig gestellt. Dabei wird zunächst offen gelassen, welche Sensoren im Endgerät in dem sogenannten „Awareness Device“ verbaut werden [SBG99]. Ein Prototyp, der in einem Mobiltelefon der Nokia 6110-6150 Serie verbaut wurde, ist in Abbildung 2.6 zu sehen.

Abbildung 2.6.: TEA (Quelle: vgl. [Lae01])

In diesem Prototyp kommen zwei Fotodioden - z.B. zur Bestimmung der Außenhelligkeit -, zwei Mikrofone - z.B. um die Lautstärke an die Umgebung anzupassen -, ein Beschleunigungssensor - z.B. zur Bestimmung der momentanen Geschwindigkeit -, ein Temperatursensor - z.B. um die Temperatur zu messen - und ein Berührungssensor - z.B. als eine einfache Eingabemöglichkeit - zum Einsatz. [Lae01]

Auch wenn das System eher einen theoretischen Ansatz liefert, um die Umgebung in der eine Anwendung zum Einsatz kommt noch genauer ermitteln zu können, und daher nicht bestimmt werden kann, ob sich das System für mobile ortsbasierte Spiele eignet, so eröffnen die zusätzlichen Sensoren vollkommen neue Möglichkeiten in Spielen. Beispielsweise erschien 2003 der erste Teil der Boktai Trilogie für den GBA. In diesen Spielen schlüpft der Spieler in die Rolle eines Vampirjägers, der als Munition Sonnenlicht benötigt. Dieses wird aber nicht in der virtuellen Welt eingesammelt, sondern durch einen Sensor, der im Spielmodul verbaut ist. Für ein erfolgreiches Spiel, ist der Spieler demnach gezwungen sich in regelmäßigen Abständen an sonnigen Plätzen aufzuhalten. Dadurch wird das Spiel noch realistischer - insofern man in diesem in diesem Zusammenhang von Realismus sprechen kann - da man nachts seinen Gegnern unterlegen ist, während man bei Tageslicht im Vorteil ist. [Wik08c] So können vollkommen neue Taktiken in die Spiele einfließen und die denkbaren Einsatzmöglichkeiten sind praktisch grenzenlos.

2.3.6. NEXUS

Auch an der Universität Stuttgart gibt es Projekte, die sich mit mobilen ortsbasierten Anwendungen beschäftigen. Das NEXUS System, das seit 1999 entwickelt wird, realisiert allerdings nicht, wie bei den anderen oben genannten Arbeiten, eine spezielle Anwendung, sondern stellt eine Plattform für diese bereit. Dabei können jederzeit sowohl neue Anwendungen, also auch neue Datenquellen eingebunden werden. Als Vorbild gilt das World Wide Web, bei dem die Informationen ebenfalls auf vielen verschiedenen Servern verteilt liegen und erst in der Anwendung zusammengefasst werden. Die Anwendungsfälle unterscheiden sich dabei nicht von den der bereits erwähnten Systemen. Somit

könnte ein Reiseführer, ein Navigationsgerät, usw. problemlos implementiert werden. Im Gegensatz zu den anderen Systemen, müssten die Daten allerdings nicht aus einer einzigen großen Datenquelle stammen. Es wäre möglich, die Aufgaben des Systems anhand der Position des Nutzers automatisch zu verändern. Beispielsweise könnte es auf Bahnhöfen als elektronische Fahrplanauskunft, gespeist von einer Datenbank der DB, in Museen als Führung, die vom jeweiligen Museum bereitgestellt wird, und in Städten als Navigationsgerät, mit Karten von der Stadtverwaltung, fungieren. Die neXus Plattform agiert dabei als Middleware mit Schnittstellen für Datenquellen und Anwendungen. [NGS+01] Die Architektur des neXus Systems ist in Abbildung 2.7 dargestellt. Das Herzstück ist dabei die sogenannte Federation Schicht. Informationsanfragen der Anwendungen gehen hier ein und werden auch zentral von hier aus beantwortet. Dazu werden Suchanfragen an beliebig viele externe Quellen mit kontextbezogenen Informationsdaten gestellt. Welche Quellen für die benötigte Information zur Verfügung stehen, wird im Area Service Register hinterlegt. Zwischen der eigentlichen Anwendung und der Föderationsschicht können weiter Services die Informationen anreichern und aufwerten. Auch ein direkter Informationszugriff, also vollkommen ohne Beteiligung der neXus Plattform, z.B. auf Daten, die im Internet angeboten werden, ist möglich. Wie man sieht, ist für das neXus Systems nicht einmal die Bezugsquelle der Positionsdaten von Bedeutung. Somit ist es auf keine bestimmte Hardware festgelegt.

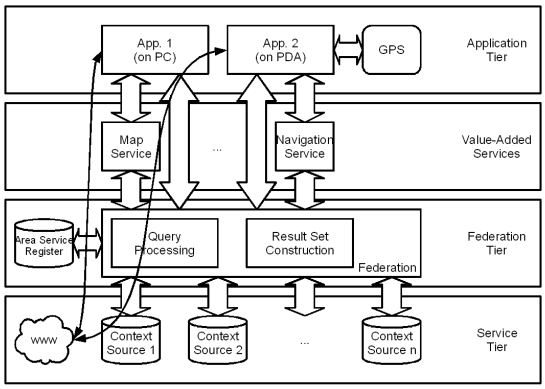

Abbildung 2.7.: neXus Architektur (Quelle: vgl. [NGS+01], [Uni07a] und [Uni07b])

Auf Grund der Architektur der neXus Plattform, die nicht auf eine Anwendung ausgelegt ist, macht eine Bewertung, ob darauf Spiele implementiert werden können oder nicht, keinen großen Sinn. Allerdings eignet sich neXus dennoch, um die unterschiedlichsten mobilen ortsbasierten Spiele zu unterstützen[1]. So könnten, abhängig davon welche Kontextinformationen zur Verfügung stehen, verschiedene Spielmodi angeboten werden, deren Kommunikation über nur eine Schnittstelle abläuft, ohne dass der Nutzer dabei auf zusätzliche Features, wie dem bewussten Zugriff auf bestimmte Informationen aus dem Internet, muss.

2.3.7. Zusammenfassung

Die in den vorangegangenen Abschnitten vorgestellten Projekte sollen einen breiten Überblick über die technische und konzeptuelle Entwicklungen von 1955 bis heute im Bereich der lokalen ortsbasierten Anwendungen liefern. Dabei handelt es sich nur um einen kleinen Ausschnitt aller

[1] beispielsweise das Studienprojekt „neXus-Ralley“

vorhandener Systeme. Diese wurden beispielhaft als Repräsentanten für Projektgruppen, die eine ähnliche Technik verwenden, gewählt. Dabei wurden sowohl Anwendungen, die mehr oder weniger reine Forschungsobjekte sind, als auch rein kommerzielle Produkte bedacht.

Wie schnell auffällt unterscheiden sich diese Projekte - natürlich neben dem Anwendungsgebiet - in einigen für mobile ortsbasierte Anwendungen entscheidenden Punkten:

1. **Ermittlung des Kontextes**: Der Großteil der Anwendungen bezieht nur die Position des Nutzers oder maximal noch einige vom Nutzer erhaltene Informationen (vgl. GUIDE, Abschnitt 2.3.3) in die Ermittlung des Kontextes mit ein. Dabei stellt [Sch02] klar, dass nur durch die Kombination mehrerer Sensoren ein exaktes Abbild der Umgebung des Nutzers erstellt werden kann. Die Wahrscheinlichkeit, den Kontext - und damit das aktuelle Einsatzgebiet der Anwendung - vorhersagen zu können, steigt mit der Genauigkeit dieses Abbildes erheblich. Der Einsatz eines Frameworks, wie TEA, würde die Qualität der meisten mobilen ortsbasierten Anwendungen steigern.

   Aber auch hinsichtlich der verwendeten Technik, um die Position des Nutzers zu ermitteln unterscheiden sich die Projekte stark. Während einige auf (etwas ungenauere) Techniken, wie IR-Sender, setzen, die die Position des Nutzers nur dann ermitteln können, wenn das Gerät in den Empfangsbereich eines Empfängers kommt, nutzen andere ein GPS-Modul. Mit GPS kann zwar die Position genauer bestimmt werden, allerdings nur, wenn das Modul empfangsbereit ist, sprich der Nutzer sich nicht innerhalb eines abgeschirmten Bereichs befindet. Daher muss bei Projekten dieser Art im Vorfeld geklärt werden, ob sich das Einsatzgebiet in einem Gebäude oder einem stark begrenzten Gebiet, das mit ausreichend IR-Empfängern ausgestattet werden kann, befindet, oder ob es sich um eine reine Outdoor Anwendung handelt, die dafür in einem (theoretisch) weltweiten Gebiet funktionstüchtig ist. Obgleich eine Kombination der beiden Systeme wünschenswert wäre, um eine optimale Ortung in Gebäuden und im Freien zu ermöglichen, findet man eine solche in keinem der „großen“ kommerziellen Systemen. [PCB00]

2. **Speicherung der Daten**: Die meisten der vorgestellten Systeme beziehen alle Informationen von externen Quellen - seien es Informationen aus dem www, oder Daten eines dem Projekt zugehörigen Servers - anstelle sie auf dem Gerät selbst zu speichern. Dadurch können Werte zentral verändert und direkt an alle Systeme verteilt werden. Ist ein Empfang allerdings kurzzeitig nicht möglich - beispielsweise, wenn sich der Nutzer außerhalb der Reichweite des WLANs aufhält -, so ist der Betrieb der Anwendung mindestens eingeschränkt, wenn nicht sogar vollkommen unmöglich. Da die Preise für Speichermöglichkeiten immer weiter fallen und die Kapazitäten dieser gleichzeitig ansteigen, wäre es sinnvoll, wenn zumindest ein Teil der Informationen auf dem Gerät selbst gehalten wird und nur veränderte Daten bei Bedarf von der externen Quelle bezogen werden.

3. **Einbezug externer Quellen**: Auch wenn die vorgestellten Systeme in erster Linie ihre Daten von einer externen Quelle beziehen, so stammen diese meist nur von einem einzigen, zentralen Server. Bei Projekten, die stark räumlich eingeschränkt sind, macht dies auch keinen großen Unterschied, aber der Datendurchsatz weitläufiger Systeme mit vielen Nutzern und vielen verschiedenen Informationsbereichen, wird darunter vermutlich leiden. Auch ist eine Skalierbarkeit bei einer schwankenden Anwenderzahl nicht gegeben. Neben den Leistungsmerkmalen, kann es auch zu Problemen bei der Datenkonsistenz kommen, da nach einer Datenänderung in der ursprünglichen Quelle die zentralen Server aller Anwendungen das gleiche Update ausführen müssen. Bei diesen Anpassungen der Werte kann es sehr leicht zu Verzögerungen kommen, wodurch der Nutzer veraltete, unnütze Informationen erhält, oder zu Fehler beim Übertragen, was die Informationen sogar falsch werden lässt. Je nach Anwendung kann dies nicht nur ärgerlich, sondern auch gefährlich sein.

Die Einbeziehung eines Frameworks wie NEXUS könnte nicht nur der Aufwand der Programmierung mobiler ortsbasierter Software erheblich vereinfacht werden, da Informationsdaten aus anderen Quellen nicht aufwendig umgewandelt in des neue System eingebracht werden müssen, sondern direkt von der fremden Quelle abgerufen werden. Auch die Zusammenarbeit mehrerer Anwendungen wird erleichtert - wenn nicht überhaupt erst ermöglicht - da dafür eine gemeinsame Schnittstelle von der Middleware bereit gestellt wird. [HKL+99] Messungen haben ergeben, dass die Leistung eines mobilen ortsbasierten Systems am größten ist, wenn - einige Annahmen bezüglich der zu Grunde liegenden Anwendung als gegeben voraussetzt - die Daten auf mehrere Server verteilt, die jeweils ein Teil des gesamten abzudeckenden Gebiets verwalten. Diese Technik wird bei der NEXUS Plattform verwendet, weshalb sich diese für den hochperformanten Einsatz einer umfangreichen Anwendung mit vielen Nutzern - wie beispielsweise einem Spiel, dessen Hauptaugenmerk auf dem Mehrspielermodus liegt - empfiehlt. [KLRS99]

System	Technik	Einsatzgebiet	Spieleplattform
Thorp Computer	keine	Roulette	ungeeignet
Active Badge	IR-Sender	Zugangskontrolle, Ortung	ungeeignet
PARCTAB	IR-Sender, Empfänger, Eingabemöglichkeit	Ortung, PDA	bedingt geeignet
GUIDE	zell-basiert	Stadtführungen	geeignet
Navicore	GPS	Navigationssystem	gut geeignet
TEA	neue Sensoren	Framework	Kontexterfassung
NEXUS	frei wählbar	Framework	unterstützend

Tabelle 2.1.: Zusammenfassung der vorgestellten mobilen ortsbasierten Anwendungen

In der Tabelle 2.1 wurden die Fakten, die für Spiele am wichtigsten sind, für jedes der sieben Projekte zusammengefasst. Neben dem Namen des Systems findet man die verwendete Technik zur Erfassung des Anwendungskontextes, das eigentliche Einsatzgebiet der Anwendung und wie sich diese Plattform sich als Grundlage für Spiele eignet.

2.4. Mobile ortsbasierte Spiele

Nachdem nun ein Überblick über einige wichtige Entwicklungen im Bereich der mobilen ortsbasierten Anwendungen berichtet wurde, soll nun auf einen Kernbereich dieser Arbeit, die mobilen ortsbasierten Spiele, eingegangen werden.

Bei den Pervasive Games muss das Spielfeld in die alltägliche Umgebung eingebunden werden. Dabei können alle dort aufgefundenen Objekte in das Spiel mit einbezogen werden. So werden alle angetroffenen Mitmenschen in die erschaffene Welt miteinbezogen - sei es als freiwillige Mitspieler oder als unbeteiligte Zuschauer. Unter diese Definition würden auch Spiele vollkommen ohne Computerunterstützung wie Parcours, bei dem vorgegebene Routen durch eine Stadt zu Fuß auf möglichst spektakuläre Weise zurückgelegt werden müssen, fallen. [Zha06] In dieser Arbeit sollen aber nur mobile ortsbasierte Spiele untersucht werden, die durch den Einsatz von technischen Hilfsmitteln die Position des Spielers ermitteln und diese Daten im Programmablauf berücksichtigen. Dabei kann es sich sowohl um mobile Spiele handeln, bei denen lediglich die Position relativ zu

anderen Elementen der Spielwelt benötigt wird, oder um Spiele, bei denen die vollständige, möglichst exakte, geografische Position zum Einsatz kommt [NPM01].

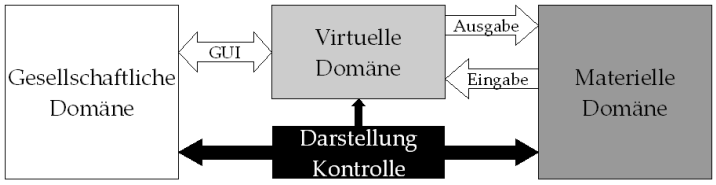

Abbildung 2.8.: Modell von Pervasive Games (Quelle: vgl. [MEM04])

Dadurch tangieren diese Spiele zwei unterschiedliche Domänen. Neben der aus normalen Computerspielen bekannten virtuellen Domäne, in der die Spiellogik abläuft und alle nicht real vorhandenen Objekte erschaffen werden, existiert hier ebenfalls eine materielle Domäne. Demnach muss die Umwelt des Spielers in des Spiel eingebunden werden und auf Veränderungen in dieser - beispielsweise wenn ein programmrelevanter Gegenstand eingesammelt wird - eingegangen werden. Meist erfolgt die Eingabe in der materiellen Domäne, während die Ausgabe in der virtuellen erzeugt wird. Wie bereits beschrieben, laufen Pervasive Games in der Regel nicht unbemerkt von den Mitmenschen ab. Diese können ebenfalls bewusst in das Spiel eingebunden werden. Ist dies der Fall, so muss auch eine gesellschaftliche Domäne eingebunden werden. In dieser werden die Kontakte mit den Mitspielern verwaltet und fließen über ein GUI der virtuellen Domäne ins Spiel ein. [MEM04] Die Zusammenhänge der drei Domänen werden in Abbildung 2.8 dargestellt.

Im Folgenden werden zwölf verschiedene Spiele vorgestellt und anschließend miteinander verglichen.

2.4.1. AR² Hockey

Bei Airhockey handelt es sich um ein Geschicklichkeitsspiel für zwei Personen. Jeder Spieler versucht mit einem Schläger einen Puck über einen Tisch in ein Tor auf der anderen Tischseite zu schieben, während die Aufgabe des anderen darin besteht dies zu verhindern. Der Puck gleitet dabei auf einem vom Spieltisch erzeugten Luftkissen und kann daher sehr hohe Geschwindigkeiten erreichen. Wird er unsauber abgewehrt, so kann er in Trudeln geraten und zu einem sehr gefährlichen Geschoss werden. [Wiko8a]

Bei AR² Hockey besteht diese Gefahr nicht. Während es sich bei dem Tisch und den Schlägern um reale Gegenstände handelt, ist der Puck lediglich ein virtuelles Objekt. Auch wenn das eigentliche Spiel sehr einfach klingt, so ist der technische Aufwand für dieses Projekt dennoch sehr groß. Die zum Einsatz kommende Hardware muss es mindestens zwei Spielern ermöglichen das exakt selbe Spielfeld zur gleichen Zeit sehen zu lassen und die Manipulationen des einen Spielers an der virtuellen Umgebung unmittelbar für den anderen sichtbar zu machen. Erschwerend kommt hinzu, dass das System, bedingt durch die hohe Geschwindigkeit des Pucks, sehr kurze Reaktionszeiten haben muss. Schließlich dürfen bei der Positionsbestimmung des Puck minimal sein, um ein realistisches Verhalten ermöglichen zu können, wenn er von einem Schläger oder der Wand abprallt. [SOYT99] Ein möglicher Aufbau des Tischs wird in Abbildung 2.9 dargestellt, in dem die wichtigsten technischen Elementen eingezeichnet sind.

Zunächst benötigt jeder Spieler ein Head Mounted Display (HMD), das allerdings auch transparent ist, damit beispielsweise der Tisch auch weiterhin zu sehen ist. Obwohl in diesen HMDs die Blickrichtung des Spielers bereits mit einem magnetischen Sensor ermittelt werden kann, wird bei AR² Hockey

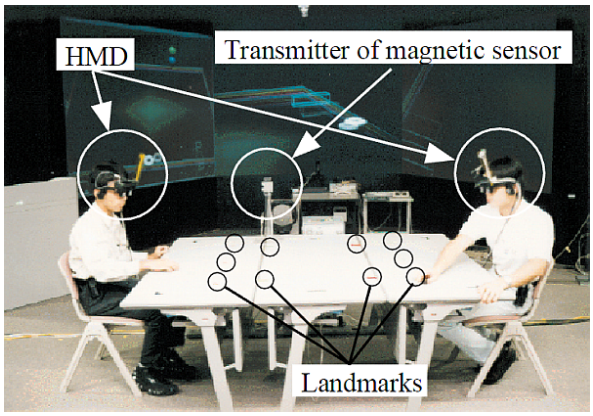

Abbildung 2.9.: Spieltisch von AR² Hockey (Quelle: [SOYT99])

noch eine zusätzliche Kamera benötigt, da das Verfahren der HMDs zu ungenau ist. Die Kamera kann diese Ungenauigkeit ausgleichen, indem sie die Markierungen auf dem Tisch erfasst und damit den exakte Blickwinkel bestimmt. Eine weitere Kamera über dem Tisch überwacht mit der selben Technik die Position der Schläger, damit diese die Laufrichtung des Pucks beeinflussen können. All diese Komponenten kommunizieren mit einem zentralen Server, der neben der Positionsberechnung auch die Ton- und Bildsynthese und den vollständigen Spielablauf durchführt. Da die Spieler nur minimale Bewegungsfreiheit benötigen, kann diese Kommunikation über eine herkömmliche kabelgebundene Ethernetverbindung erfolgen.

Nach ausgiebigen Testläufen kamen die Probanden zu dem Ergebnis, dass sich das grundsätzliche Spielgefühl kaum von dem bei einer realen Airhockey-Partie unterscheidet. Allerdings ist das Sichtfeld durch das HMD zu stark eingeschränkt, so dass häufige Korrekturen der Blickrichtung nötig wurden, um den Puck nicht aus den Augen zu verlieren. [OSYT98] Dadurch ist allerdings der erste Grundsatz eines Pervasiv Systems - dem Nutzer nahezu transparent zu erscheinen - erheblich gestört.

2.4.2. ARQuake

Bei ARQuake handelt es sich um eine Erweiterung eines bestehenden, äußerst beliebten Desktop-computerspiels. Bei dem Originalspiel erlebt der Spieler die virtuelle Welt aus den Augen seines Avatars und steuert diesen mittels Maus oder Tastatur durch dunkle Gewölbe und wehrt sich gegen mehrere Monsterarten. Für die Umsetzung dieses Spiels in eine Augmented Reality kommt ein transparentes Head Mounted Display zum Einsatz, welches dem Spieler einerseits ermöglicht die reale Umgebung weiterhin sehen zu können und andererseits mittels Spiegel virtuelle Elemente, wie Gegner oder sammelbare Gegenstände, einblenden kann. [PT02] Das Spiel aus Sicht eines Nutzers ist in Abbildung 2.10 abgebildet.

Auch wenn das Spiel relativ alt ist und nicht außergewöhnlich hohe Hardwareansprüche hat, reicht ein Mobiltelefon für die pervasive Umsetzung nicht aus. Ein Linux Notebook mit GPS-Empfänger wird stattdessen benötigt. Dieses wird als Rucksack dem Spieler auf den Rücken geschnallt, um dessen Mobilität nicht unnötig einzuschränken. Eine Steuerung mit Maus und Tastatur wäre ebenfalls nicht mit dem Grundgedanken von ubiquitären Anwendungen der intuitiven Eingabe vereinbar. Daher wurde eine Spielzeugwaffe entwickelt. Das eigentliche Zielen erfolgt zwar dennoch ausschließlich über die Blickrichtung, aber die Abgabe von Schüssen kann vollkommen natürlich erfolgen und ein

Abbildung 2.10.: ARQuake (Quelle: [Wea08])

weitere an die Waffe angebrachter Knopf ermöglicht das Wechseln der Waffe. Um die Spielfigur durch die Welt zu bewegen wird keine Eingabe benötigt, da dies über ein Ortungssystem funktioniert.

Da die modellierte Welt neben Außenarealen auch Innenbereiche beinhaltet, reicht eine einzige Technik dafür leider nicht aus. Außerhalb von Gebäuden kann selbstverständlich das GPS-Signal zur Positionsbestimmung verwendet werden. Nähert man sich aber einem Bauwerk, so kann die Ungenauigkeit des Signals dafür sorgen, dass virtuelle Objekte nicht auf dem Display sichtbar sind, da sie sich mit den Mauern überlappen. Daher wird für den Bereich um Bauten herum eine GPS Ortung um die Ortung mittels Passermarken, die von einer in das HMD eingebauten Kamera erkannt werden können, erweitert, die an der Fassade angebracht werden. Innerhalb von Gebäuden kann überhaupt kein GPS-Signal empfangen werden. Daher müssen auch hier Passermarken verwendet werden. Allerdings könnten diese etwas kleiner ausfallen, als die vor dem Gebäude.

Da der Spieler nicht einfach in eine virtuelle Welt versetzt werden soll, muss eine wirklich existierende Umgebung in eine Karte für Quake umgesetzt werden, in der alle real existierenden Hindernisse mittels virtuellen äquivalenten Objekten ersetzt werden. So kann verhindert werden, dass Monster durch Wände laufen, oder Objekte, die eingesammelt werden müssen unerreichbar für den Spieler sind.

ARQuake kann demnach als eines der wenigen echtes ubiquitären Systeme in einer Augmented Reality angesehen werden. [TCD+02]

2.4.3. Augmented Knight's Castle

Während die meisten Computerspiele dem Spieler nur die Möglichkeit geben eine vorgegebene Geschichte zu erleben, und dieser nur minimal Einfluss auf die Handlung hat, so schlägt das „Augmented Knight's Castle“ eine vollkommen andere Richtung ein. Für Kinder soll hier der Umgang mit normalem Spielzeug interessanter werden, indem die Geschichten, die sie sich ausdenken und immer wieder neu gestalten können um multimediale Elemente angereichert werden. Als Plattform dient das Knights Empire Castle von Playmobile, eine mittelalterliche Burg, die um viele weitere, zum Setting passende Elemente, wie Ritter, Gebäude, usw. erweitert werden kann. In Abbildung 2.11 ist eine mögliche Spielumgebung aufgebaut.

Abbildung 2.11.: Aufbau der großen Königsritterburg (Quelle: [LHB06])

Damit auf Aktionen der Kinder entsprechend reagiert werden kann, müssen alle Spielfiguren nicht nur unterschieden, sondern auch deren Position bestimmt werden. Um, wie bei der Definition eines pervasiven Systems gefordert, den Nutzer in seinen Aktionen nicht einzuschränken, muss eine Technik gewählt werden, die problemlos und nahezu unsichtbar in die Umgebung verbaut werden kann. Im Fall des Augmented Knight's Castle fiel die Wahl dabei auf RFID, um jedem Gegenstand der Spielwelt eine eindeutige ID zu vergeben. Auch die Ortung der Gegenstände kann mittels RFID erfolgen, da die Reichweite der Signale sehr gering ist. Daher genügt es, in jedem Raum eine Antenne aufzubauen und wenn die ID eines bekannten Gegenstandes von dieser erkannt wird, so befindet sich der Gegenstand in diesem Raum.

Ein Spielserver hat, wenn die Positionsdaten übermittelt wurden die Möglichkeit, auf bestimmte Ereignisse zu reagieren. Beispielsweise könnte automatisch eine Fanfare eingespielt werden, wenn der König eine Szene betritt. Zusätzlich können auch zufällige Ereignisse ausgelöst werden, um einen dynamische und lebendige Umwelt zu vermitteln. So wäre bellende Hunde im Dorf oder zwitschernde Vögel im Wald denkbar. Auch wenn diese Ereignisse im Vorfeld definiert werden müssen, so schränkt dies die Fantasie der Kinder dennoch nicht ein, da sie völlig frei entscheiden können wann und ob sie ausgelöst werden sollen.

Neben rein akustischer Untermalung ermöglicht die RFID Technologie aber noch mehr Unterstützung beim Spiel. Mit einem tragbaren RFID-Lesegerät - beispielsweise ein herkömmliches Mobiltelefon mit einem entsprechenden Adapter - kann die Spielwelt um weitere virtuelle Inhalte erweitert werden. So könnten Hintergrundinformationen zu den einzelnen Figuren gegeben werden, um deren Charakter mehr Tiefgang zu verleihen. Es wären aber auch gerenderte Videos vorstellbar, die den Ausgang von Kämpfen zwischen zwei bestimmten Ritter simulieren. Die Entscheidung einer Schlacht kann entweder zufällig erfolgen, oder den Hintergrund der Spielfigur, wie bei einem Rollenspiel, miteinbeziehen. Da die Zielgruppe dieses Spiels in erster Linie Kinder sind, wäre es wünschenswert das Lesegerät nicht in einem Mobiltelefon zu verbauen, sondern in einem Objekt, das in die Spielumgebung passt. Im Fall der großen Königsritterburg wäre ein Zauberstab denkbar, der den Kindern die Macht verleiht den Figuren Leben einzuhauchen. [LHB06]

2.4.4. Bot Fighters!

Bot Fighters! ist das erste kommerzielle Pervasive Game für Mobiltelefone. Das 2001 von der schwedischen Softwarefirma „It's Alive" veröffentlichte Spiel wird heute von verschiedenen Providern in mehreren Ländern in Europa angeboten. Der Inhalt ist dabei recht simpel. Jedem Spieler steht ein Kampfroboter zur Verfügung mit dem versucht werden muss, andere, feindliche Roboter zu finden und anschließend zu zerstören.

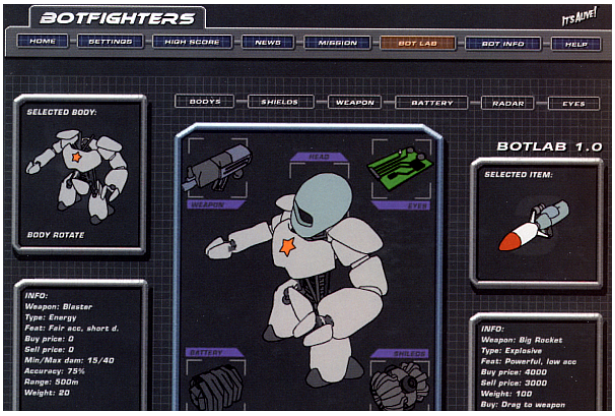

Abbildung 2.12.: Webinterface von Bot Fighters! (Quelle: [It'02])

Das Programm setzt sich aus zwei unterschiedlichen Produkten, einem Webinterface und dem eigentlichen Handyspiel, zusammen. Das in Abbildung 2.12 dargestellte Webinterface wird benötigt, um neue Spieler anzumelden, den Avatar des Spielers zu erstellen und neue Ausrüstung zu kaufen. Der Hauptteil des eigentlichen Spiels kommt ohne Grafische Ausgabe aus und läuft auf einem normalen Mobiltelefon ausschließlich im Textmodus ab. Dem Spieler stehen eine begrenzte Anzahl an Kommandos zur Verfügung, die er per kostenpflichtiger SMS Nachricht an den Spielserver ausführen kann. So kann beispielsweise nach anderen Spielern in der Umgebung gesucht oder ein Kampf eingeleitet werden, sollte sich ein Gegner in der Nähe aufhalten. Beginnt ein Kampf, so läuft dieser auch im Textmodus, allerdings mit anderen Befehlen, wie Flucht oder Angriff, ab. Auf jede Nachricht erhält der Spieler umgehend eine Rückmeldung vom System. [Soto2]

Bedingt durch die einfache Natur des Spiel wird neben einem herkömmlichen GSM-Mobiltelefon keine weiter spezielle Hardware benötigt. Die Positionsdaten der Spieler wird über das GSM-Funknetz ermittelt. Dadurch wird zwar kein GPS-Empfänger benötigt, aber die Genauigkeit dieses Verfahrens hängt stark von der Größe der Funkzelle ab und ist in der Regel deutlich ungenauer als die Ermittlung per GPS. Da allerdings nur die Position relativ zu anderen, ebenfalls ungenau georteten Spielern benötigt wird, ist diese Technik für das Spiel ausreichend. Diese geringen Hardwareanforderungen werden benötigt, um eine möglichst große Verbreitung zu ermöglichen, wie sie für kommerzielle Produkte dringend erforderlich sind. [Napo6]

Mit Bot Fighters! sollten - neben einem möglichst hohen Umsatz - laut [Soto2] drei Ziele erfüllt werden:

1. Im Gegensatz zu anderen Pervasive Games, wie Can You See Me Now?, sollte es theoretisch jederzeit und überall möglich sein zu spielen.
2. Das Spiel sollte an die mobilen Geräte keine hohen Anforderungen stellen und versuchen möglichst kompatibel zu anderen Systemen sein.

3. Mit der Verbindung von realen Orten und Personen mit dem Spiel sollte ein soziales Gefüge geschaffen werden.

Diese Punkte wurden mehr oder weniger gut erfüllt. Bot Fighters! lässt sich zwar immer spielen, wenn sich mindestens ein weiterer Spieler in der näheren Umgebung befindet, aber ausschließlich in Gebieten, in denen das Spiel angeboten wird und nur bei einer bestehenden Verbindung zu einem GSM-Funknetz. Durch den konsequenten Einsatz von Java bei der Programmierung reicht ein Mobiltelefon mit J2ME Unterstützung aus. Die schlichte Grafik stellt auch keine Anforderungen an das Display, so dass auch monochrome Anzeigegeräte verwendet werden können. [Nap06] Auch die sozialen Aspekte werden recht gut erfüllt, da zum einen neue anonyme Personen zunächst im Spiel, dann aber auch in der Realität kennen gelernt werden können und zum anderen jeder Spieler sich ein virtuelles Ansehen schaffen kann - beispielsweise durch einen guten Highscore oder durch eine starke Ausrüstung des eigenen Roboters - der u.U. stark von seinem realen sozialen Status abweicht [Soto2].

2.4.5. Can You See Me Now?

Eine pervasive Umsetzung des bekannten Kinderspiels „Fangen“ stellt „Can You See Me Now?“ dar, das von Blast Theory in Zusammenarbeit mit dem Mixed Reality Lab der Universität von Nottingham entwickelt wurde. Das Besondere an CYSMN ist, dass sich nur die Fänger, die im Spiel als „Runner“ bezeichnet werden, durch einen im Vorfeld begrenzten Bereich der realen Welt bewegen. Die restlichen Spieler steuern lediglich einen Avatar durch ein virtuelles Abbild des selben Gebiets. Die Positionsdaten dieser virtuellen Spielfiguren werden mittels einer Internetverbindung sowohl an alle Mitspieler, als auch auf die mobilen Computer, die jedem Runner vor Spielbeginn ausgehändigt werden, gesendet. Die Onlinespieler „sehen“ die Runner dank eines GPS Empfänger, der in deren tragbaren Computer verbaut ist. [BML05]

Zu Spielbeginn können sich Onlinespieler auf der Webseite anmelden und werden in einer Warteliste eingereiht. Die ersten fünfzehn Spieler betreten daraufhin die virtuelle Welt und bekommen eine Karte mit allen Mitspielern, Gegenspielern und relevanten Punkten angezeigt, wie es in Abbildung 2.13 zu sehen ist. Dabei handelt es sich allerdings nur um eine statische Abbildung ohne jegliche bewegte Elemente, wie Autos oder Passanten. Innerhalb des Spielgebiets dürfen sich die Spieler zwar frei bewegen, aber es dürfen dabei keine Gebäude betreten oder physische Hindernisse überschritten werden. Während der Spielzeit kann mit den anderen Spielern über Textnachrichten kommuniziert werden, um so Taktiken zu erarbeiten, um den Runnern zu entkommen. Diese vier Fänger müssen sich zu Spielbeginn nicht anmelden, da es sich bei diesen um Angestellte des Veranstalters von CYSMN handelt. Sie verteilen sich bereits vor Beginn des Spiels im realen Spielgebiet und müssen versuchen, sobald der erste Spieler die Welt betreten hat, diesen „zu sehen“, was bedeutet, dass sie sich dem Avatar auf fünf Meter nähern können. Wurde ein Onlinespieler gesehen, so muss dieser das Spiel verlassen und der nächste der Warteliste rückt für ihn nach. Die Zeit zwischen Betreten und Verlassen der virtuelle Welt wird vom System aufgezeichnet, damit die Leitung der Spieler miteinander verglichen werden kann und sich ein Sieger ermitteln lässt. [BCF+06]

Der Ausfall des GPS Signals innerhalb von Gebäuden und die Ungenauigkeit in dicht besiedelten Gebieten stellen die technischen Hauptprobleme bei CYSMN dar. Um mit den Ausfällen in geschlossenen Räumen umgehen zu können, wurden diese für die Onlinespieler gesperrt. Die Ungenauigkeit kann allerdings nicht ausgeglichen werden. Dabei ergaben Auswertungen des System-Logs, dass dieser Fehler im Mittel eine Abweichung von zwölf Meter ausmacht. Dies bedeutet für die Runner, dass ihre virtuelle Position nicht mit ihrer realen übereinstimmt und dass das „Fangen“ erheblich erschwert wird. Für die Onlinespieler ist dieser Fehler allerdings praktisch unbemerkbar. Daher sind die Reaktionen auf CYSMN trotz dieses Problems überwiegend positiv und das Potenzial dieses

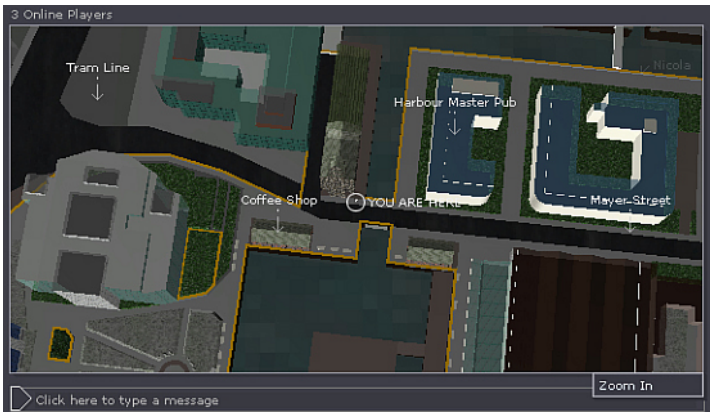

Abbildung 2.13.: Interface eines CYSMN-Onlinespielers (Quelle: [Rus07])

Ansatzes die virtuelle Welt mit der realen zu verbinden wurde von allen Beteiligten als sehr hoch betrachtet. [FBA+03]

Da die Runner allerdings ausschließlich aus professionellen Darstellern bestehen, stellt CYSMN für die Mitspieler nur ein stark eingeschränktes pervasives Spielerlebnis dar, da diese weder mobil sind, noch ortsabhängig agieren können. Außerdem ist der benötigte Aufwand für eine Partie recht groß und daher kann Can You See Me Now? nur zu festgelegten Zeiten an bestimmten Orten ausgeübt werden. Seit 2001 fand das Spiel aus diesem Grund lediglich in Sheffield, Rotterdam, Oldenburg, Köln, Brighton, Barcelona, Tokyo, Banff, Chicago, Dublin, Krems und Preston statt. [BM08a]

2.4.6. Epidemic Menace

Das Fraunhofer-Institut für Angewandte Informationstechnik will mit einem Pervasive Game die Grenzen herkömmlicher Spiele aufbrechen, indem diese um verschiedene elektronische Medien, wie Musikuntermalung oder Videoeinspielern, erweitert werden, die dynamisch auf die jeweilige Spielsituation reagieren können. Mit diesem Ansatz wurde in Zusammenarbeit mit verschiedenen anderen Partnern Epidemic Menace im Rahmen des EU-Projekts IPerG entwickelt. Das Spiel konnte bereits 2005 von freiwilligen Testern erprobt werden. [Dee05]

Zu Anfang wird den Spielern in einem kurzen Video das Setting näher gebracht. Sie schlüpfen in die Rolle von Virenexperten, die freigesetzte Viren in einem abgesteckten Bereich innerhalb von drei Stunden zu bekämpfen. Dabei wird aber nicht kompetitiv vorgegangen, sondern nur durch ein gutes Zusammenspiel aller Beteiligter kann die Mission Erfolg haben. Die Spieler werden dazu in zwei Teams eingeteilt, von denen das eine zur Virenbekämpfung und das andere als Einsatzleitung eingesetzt wird. Trotz dieses gemeinschaftlich Charakters des Spiels existiert ein Team-interner Highscore, so dass am Ende der beste Spieler ermittelt werden kann, um damit die Einsatzbereitschaft zu erhöhen. Die Aufgaben und damit die benötigte Ausrüstung unterscheidet sich bei diesen zwei Gruppen sehr. Die Einsatzleitung, die während der vollständigen Spielzeit in einer Kommandozentrale bleibt, verfolgt das Vorgehen der restlichen Spieler und versucht Hintergründe zum Ausbruch der Viren aufzudecken. Dazu steht ihnen stationäre Computer zur Verfügung, an denen sie die entdeckten Viren untersuchen können. Außerdem können sie die Position ihrer Teammitglieder daran sehen und jederzeit Kontakt mit diesen aufnehmen. Die Außenteams können zwischen mehreren Hilfsmitteln zur Bekämpfung der Viren wählen. Zu diesen Hilfsmitteln zählt ein HMD, auf dem die Viren

Abbildung 2.14.: Ausrüstung für Epidemic Menace (Quelle: [Dee06])

direkt eingeblendet werden, ein Mobiltelefon, das gefundene Viren auf einer Karte anzeigt und einfangen kann und einem Kopfhörer, über den Kontakt zur Einsatzleitung besteht und auf dem Musik abhängig von der Nähe der Spieler zu einer Virenquelle abgespielt wird. Unabhängig von dem gewählten Hilfsmittel bekommen alle Mitglieder eines Außenteams ein Gerät, das die Ortung der Spieler ermöglicht. [LOPBG07] In Abbildung 2.14 ist ein vollständig ausgerüsteter Spieler und einige virtuelle Spielelemente zu sehen. Jedoch können auch die Viren auf ein verändertes Umfeld, wie die aktuelle Wetterlage, reagieren, um das System noch lebendiger und realer erscheinen zu lassen [Dee05].

Das Ortungsgerät wurde mit einem GPS Empfänger, der über Bluetooth mit einem PDA verbunden ist. Dieser übermittelt die Daten über einen WLAN Zugang an einen zentralen Server. Auf diesem Server werden die Daten verwaltet und das Verhalten der Viren gesteuert. Die errechneten Informationen werden an alle Clients, wie den Rechnern im Kontrollzentrum oder den Smartphones und den HMDs, weitergeleitet. [LOPBG07]

Obwohl sich die Probespieler sehr begeistert von Epidemic Menace zeigten [Dee06], so ist die Auswertung in den Bereichen Wirtschaftlichkeit, Ausführung und technische Unterstützung sehr ernüchternd. Da die Wiederspielbarkeit aufgrund der starren Hintergrundgeschichte sehr gering, die Kosten zur Realisierung aber sehr hoch sind, müsste ein riesiges Kundeninteresse erzeugt werden, um kommerziell erfolgreich sein zu können. Bei den ersten Tests konnte auch kein flüssiger Spielfluss aufkommen, da die Spielleitung zu oft Eingreifen musste, um den Umgang der zu komplizierten Technik zu erklären. Der Spielfluss kann ebenfalls ins stocken geraten, wenn die eingesetzte Technik ausfällt. [FLS06] Allerdings kam es gerade bei der Konnektivität der PDAs zum Server trotz einer eigentlich sehr guten WLAN Abdeckung zu häufigen Verbindungsausfällen [LOPBG07].

2.4.7. Human Pacman

Auch wenn das 1980 erschienenen Pucman, das erst später in Pac-Man umbenannt wurde, nicht das erste Computerspiel der Welt war, so gilt es dennoch bis heute als einer der bekanntesten Videospiele aller Zeiten. Auch wenn das Spiel denkbar einfach ist - der Spieler „frisst“ in einem Labyrinth unter Zeitdruck Punkte, während er von „Geistern“ verfolgt wird - so erschienen auf Grund dieser Beliebtheit dennoch viele offizielle Fortsetzungen und unzählige Klone. [HS08] In Abbildung 2.15 ist

das Spielfeld des Originals zu sehen, das auch meist für die Nachfolger unverändert übernommen wurde.

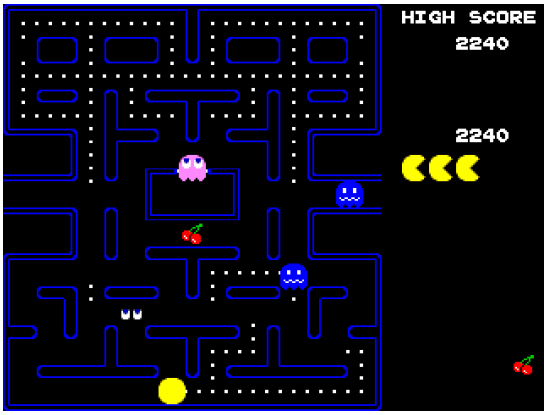

Abbildung 2.15.: Screenshot des original Pucman (Quelle: [Bur07])

Das Mixed Reality Lab der National University of Singapore setzte dieses beliebte Spielprinzip mit Hilfe neuer mobiler Technik als Pervasive Game möglichst nahe am Original um. Dabei wurde auf die heutigen Wünsche der Spieler nach physischer Interaktionsmöglichkeiten und einer virtuellen dreidimensionalen Spielwelt eingegangen. Während Computerspiele lange Zeit alleine und abseits der Außenwelt stattfanden, so sollte bei Human Pacman auch soziale Aspekte Beachtung finden. [Mix08]

Im Gegensatz zum Original nehmen bei Human Pacman die Spieler sowohl die Rolle des Pacmans, als auch die der Geister ein. Alle angemeldeten Spieler werden in eines der beiden Teams eingeteilt und bewegen sich gleichzeitig über das Spielfeld. Von einem begrenzten Gebiet wird ein Modell angelegt und darin werden neben virtuelle Marken, die von den Pacman gefressen werden müssen, um zu punkten, auch reale Gegenstände verteilt. Diese Gegenstände verleihen den Pacman wenn sie eingesammelt werden für eine kurze Zeit Immunität vor den Geistern. Wenn das Spiel beginnt müssen die Pacmen versuchen so viele der Punktemarken einzusammeln, indem sie über das reale Spielfeld laufen, ohne selbst von einem Geist „gefressen“, bzw. berührt, werden. Der Spieler sieht die Marken natürlich nur auf einem Display, allerdings nicht in der Vogelansicht, sondern aus der Egopespektive, wie in Abbildung 1.1 zu sehen ist. Daraus ergibt sich das Problem, dass die Übersicht über die noch nicht eingesammelten Marken und auch die Position der Geister verloren gehen kann. Daher darf noch eine dritte Gruppe in das Spiel eingreifen. Diese Gruppe, die sogenannten Helfer, nimmt am Geschehen nur über das Internet teil und sieht das Spielfeld mit allen Positionen der Spielobjekten aus der Draufsicht. Direkt intervenieren dürfen sie zwar nicht, aber sie können ihr Wissen an einen ihnen zugeteilten Spieler - sei es ein Pacman oder ein Geist - weitergeben. [CGL+04]

Um das virtuelle Modell mit der realen Welt möglichst nahtlos ineinander übergehen zu lassen, kommen mehrere Techniken zum Einsatz. Zunächst wird für jeden Spieler ein GPS-Empfänger benötigt, um seine Position bestimmen zu können benötigt. Da das Spiel aber auch in Gebäuden funktionieren soll, wird ein sogenanntes „dead reckoning module“ verwendet, das neben dem GPS-Signal auch Kompassdaten, Schrittzähler und Höhenmesser auswerten kann. Diese Daten werden an einen Server mittels WLAN übertragen. Um das Spielerlebnis intensiver gestalten zu können werden die Spieler mit HMDs ausgestattet, die auch auf deren Blickrichtung eingehen können. Um die realen Immunitätspillen erkennen und einsammeln zu können werden diese mit Bluetooth-Chips ausgestattet, die den Spieler auf sich aufmerksam machen, wenn er sich ihnen nähert. Schließlich

Abbildung 2.16.: Bild der Human Pacman Weste (Quelle: vgl. [CFG+03])

muss noch der Server erfahren, wenn ein solcher Gegenstand oder ein Spieler gefangen wird. Dazu werden die Gegenstände und Spieler mit Drucksensoren ausgestattet, die bei einer Berührung die anderen Mitspieler und Helfer darauf aufmerksam machen können. Diese Geräte und Sensoren werden von einem tragbaren Computer gesteuert. Um die Spieler nicht allzu sehr einzuschränken, wäre eine Anbringung an einer Weste, wie in Abbildung 2.16 zu sehen ist, möglich. Die Helfer benötigen lediglich einen normalen PC mit Internetzugang. [CFG+03]

2.4.8. MOPET

Während das frühere Bild eines Computerspiele-Spielers das einer trägen, meist übergewichtigen und untrainierten Person war, so versuchen immer mehr Hersteller diese Image abzulegen und damit eine neue Generation von potenziellen Interessenten anzusprechen. Daraus entsteht neue Hard- und Software, mit der versucht wird in einem Spiel ein vollständiges Trainingsprogramm zu verpacken, wie beispielsweise bei der Wii-Fit (siehe [Nino8]). [Frio8]

Prinzipiell fallen auch alle Pervasive Games in diese Rubrik, da ein solches Spiel nur dann zu Stande kommen kann, wenn der Spieler sich selbst bewegt. Der Mobile Personal Trainer, oder kurz MOPET, geht dabei einen Schritt weiter. Ziel dieses Projekts ist es ein System zu erstellen, das

- ein körperlich anstrengendes Spiel durch Computereinsatz unterstützt,
- einen virtuellen Trainer zur Verfügung stellt und
- mobil ist, um jederzeit und überall Einsatzbereit sein zu können.

Gerade der letzte Punkt hebt dieses Produkt von den anderen Ansätzen in diesem Bereich ab. Die Idee dabei ist mit Hilfe eines PDAs einen Trimm-Dich-Pfad für den Nutzer zusammen zu stellen, der sich aus kurzen Joggingstrecken und dazwischenliegende Fitnessübungen besteht. MOPET muss dabei drei Funktionen übernehmen:

- Das System muss Aufgaben eines Navigationsgeräts übernehmen, damit sich der Spieler auch auf ihm unbekannten Parcours problemlos zurechtfinden kann.
- Das System muss den Spieler bei den Übungen überwachen und bewerten, so dass dieser motiviert ist die Trainings-Effizienz und Effektivität stetig zu steigern.

- Das System muss an den Übungsstationen dem Spieler die korrekte Ausführung der Übung präsentieren. Dabei muss darauf geachtet werden, dass der Bewegungsablauf fehlerfrei ist, um Verletzungen zu vermeiden. Die Tatsache, dass der Trainer als dreidimensionales Modell erstellt wurde und vollständig animiert wurde, kommt dieser Funktion zu gute, da so im Gegensatz zu einem statischen Video, der Blickwinkel dynamisch gewählt werden kann.

Alles was dafür neben dem PDA benötig wird ist ein GPS-Empfänger zur Ermittlung der momentanen Position des Spielers, um die Navigationsfunktion übernehmen zu können. Auf einer auf dem Display angezeigten Karte kann so neben der Route und den Stationen auch der aktuelle Fortschritt eingesehen werden. Durch eine zusätzliche, akustische Ausgabe kann in der Regel ein dauerhafter Blick auf das Gerät vermieden werden. Erreicht der Spieler eine Trainingsstation macht das System durch einen Signalton auf sich aufmerksam und blendet statt der Karte den Trainer auf dem Display ein. Der Trainer führt daraufhin die Übung exakt vor. Im Gegensatz zu herkömmlichen Trimm-Dich-Pfade, bei denen der Sportler den Bewegungsablauf anhand von kleinen - selbstverständlich statischen - Abbildungen, die leicht übersehen werden können oder vollständig fehlen, verstehen muss, wird hier versucht durch den audiovisuellen Einsatz schädliche Fehler auszuschließen. Die beiden Betriebsmodi sind in Abbildung 2.17 zu sehen.

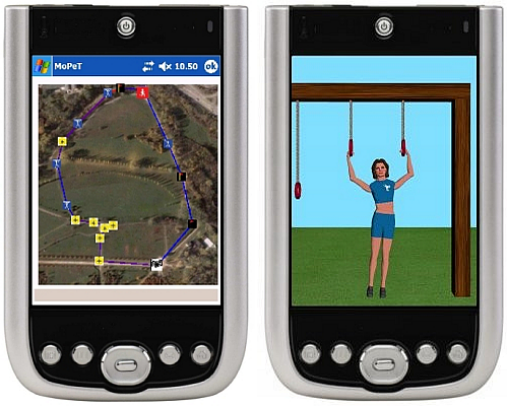

Abbildung 2.17.: Karte und Trainer von MOPET (Quelle: vgl. [BCN06b])

Zur Motivation des Spielers steht dem Trainer allerdings nur die benötigte Zeit, die für die einzelnen Joggingabschnitte benötigt wird - und damit verbunden die Laufgeschwindigkeit - zur Verfügung. So kann die Trainingsleistung nur mit Durchschnittswerten oder früheren Versuchen des selben Spielers in Bezug gesetzt, allerdings nicht auf die Leistung bei den Übungen oder den persönlichen Fitnessgrad eingegangen werden. Dieses Manko soll in späteren Versionen ausgemerzt werden, indem die Anzahl der auswertbaren Kontextinformationen - beispielsweise durch den Einsatz von Herzfrequenzmessern - erhöht wird. [BCN06a]

2.4.9. Pirates!

Auch das Nokia Research Center entwickelte in Zusammenarbeit mit Play ein Pervasive Game. Bei Pirates! handelt es sich um ein kontextsensitives mobiles Spiel, das Möglichkeiten bietet, mit anderen Spielern zu interagieren. Dabei werden mehrere Genres, wie Wirtschaftsstrategie-, Action- und Rollenspiel gemischt. Der Spieler übernimmt die Rolle eines Kapitäns auf einem Piratenschiff. Dieses ist allerdings nur sehr klein und schlecht ausgerüstet. Mit fortschreitender Spielzeit kann es

allerdings durch den Einsatz von Gold, das für das Lösen von Aufgaben oder durch Warenverkehr erlangt wird, aufgerüstet werden. Die Spielwelt besteht aus mehreren Inseln, auf denen Handel betrieben werden kann und Aufträge vergeben werden.

Zu Beginn ist dem Spieler allerdings nur eine Insel bekannt und weitere müssen zunächst gefunden werden. Hierfür werden die kontextsensitiven Daten benötigt. Das Schiff wird dadurch bewegt, indem der Spieler seine Position auch in der realen Welt verändert. Im Gegensatz zu den meisten Pervasive Games werden die exakten GPS-Positionsdaten nicht benötigt, da die entscheidenden Orte, wie Inseln, an festen Orten in der realen Welt verankert sind. An diesen Stellen wird daher lediglich ein Sensor benötigt, der erkennt, wenn sich ein Spieler mit einem entsprechend ausgerüsteten PDA diesem Punkt nähert.

Abbildung 2.18.: Bildschirmanzeige von Pirates (Quelle: vgl. [BFHL01])

Die Anforderungen an den PDA sind dabei sehr gering, da neben einem Radiofrequenz-Empfänger nur ein Display, eine einfache Eingabemöglichkeit und einen WLAN Verbindung. Die Rechenleistung ist praktisch unbedeutend, da auch die Ausgabe sehr einfach gehalten ist, wie in Abbildung 2.18 zu sehen ist. Der obere Bildschirm zeigt die Seekarte, auf der neue Inseln erforscht werden können, während die Ansicht unten die Navigation auf einer Insel zeigt. Je nachdem in welchem der beiden Modi der Spieler sich befindet, können die Sensorpunkte der realen Welt für unterschiedliche Positionen in der virtuellen Welt stehen. Da die Position nur relativ zu einem Startpunkt bestimmt wird, kann Pirates! theoretisch an jedem Punkt der Erde gespielt werden - eine WLAN Verbindung und eine Umgebung mit Sensorpunkten vorausgesetzt.

Die WLAN Verbindung wird benötigt, um Kontakt zu einem zentralen Spielserver aufbauen zu können. Auf diesem läuft nicht nur die Spiellogik ab, sondern es werden auch sämtliche Daten der Spieler hier verwaltet. Diese Daten beinhalten neben dem erwirtschafteten Gold auch Erfahrungspunkte oder der letzten ermittelten Position. Anhand dieser Daten kann auch eine Highscoretabelle erstellt werden, damit die Leistung alle Spieler vergleichbar gemacht wird. Meldet sich ein bekannter Spieler bei dem Server an, so können diese Daten an seinen Client weitergeleitet werden. [BFHL01]

Der Vergleich der erreichten Punkte stellt im Spiel allerdings nicht die einzige soziale Komponente im Spiel dar. Die in den PDAs verbauten Sensoren finden sich auch gegenseitig, so dass Spiele zwischen zwei Spielern möglich, sobald sich deren Schiffe ausreichend nahe kommen. Darauf werden deren Daten im Bildschirm angezeigt, wie man im oberen Teil von Abbildung 2.18 sehen kann. Wenn die Spieler wollen, kann dann ein Kampf begonnen werden, bei dem die Errungenschaften zum Vorteil der Spieler eingesetzt werden können. Allerdings kann das Spiel problemlos vollkommen ohne diesen

Mehrspielermodus ablaufen, wenn sich nicht ausreichend Mitspieler im Umfeld des Spielers befinden oder dieser keine Lust auf diese Duelle hat. [FLBH01]

2.4.10. REXplorer

Das REXplorer-Projekt wurde von der RWTH Aachen und der ETH Zürich in Kooperation mit dem Regensburg Experience, einem Erlebnismuseum der Stadt Regensburg, erstellt. Es soll Touristen die Möglichkeit bieten, den historischen Ortskern auf eine vollkommen neue Art zu erforschen. Das Spiel beginnt in dem Museum, indem die interessierten Besuchern mittels einer Videobotschaft in die Geschichte eingeführt werden. Diese handelt von einem real existierende Grabstein, der mit mystischen Runen verziert ist. Der Spieler soll diese Symbole entschlüsseln, indem er Hinweise verfolgt, die ihn an sehenswerte Orte in der Stadt führt. So verschmilzt in diesem Projekt Spiel und Bildung, so dass die Touristen mit möglichst viel Spaß alles Wissenswerte über die historisch bedeutsamen Plätze und Gebäude erfahren. Um diesen Service möglichst dauerhaft anbieten zu können, wurde darauf geachtet, dass es möglichst ohne zusätzliches Personal - wie beispielsweise Schauspieler, die die Geschichte vorantreiben - auskommt.

Aufträge erhält der Spieler über ein extra für diesen Zweck entwickeltes Eingabegerät, das im Museum gemietet werden kann. Der Vorteil in dieser Vorgehensweise liegt darin, dass das Zielsystem bei der Entwicklung der Software bereits bekannt ist, so dass keinerlei Abstriche gemacht werden müssen, um möglichst viele Geräte unterstützen zu können. Andererseits können die Geräte mit Funktionen und Sensoren ausgestattet werden, die für das Spiel benötigt werden. Auf diesem Eingabegerät wird eine Karte der Stadt angezeigt und wichtige, bereits besuchte Plätze werden eingezeichnet. Des Weiteren werden darin Hinweise gegeben, wohin sich der Spieler als nächstes begeben sollte. [BKB+07] In Abbildung 2.19 wird diese Karte gezeigt.

Abbildung 2.19.: REXplorer (Quelle: [Bie05])

Soweit unterscheidet sich REXplorer nur wenig von einem herkömmlichen Navigationsgerät. Die Besonderheit macht sich erst bemerkbar, wenn sich der Spieler einem wichtigen Punkt nähert. Dann signalisiert ihm das Eingabegerät - bei dem es sich um ein erweitertes Nokia Mobiltelefon handelt -, dass er an dieser Stelle mit der virtuellen Welt interagieren kann. Dies geschieht indem er „Zaubersprüche“ mit seinem „Zauberstab“ wirkt. Ein solcher Zauberspruch ist eine definierte Bewegung des Eingabegeräts durch die Luft.

Diese Gestiken werden über eine Kamera erkannt, die immerzu Bilder aufnimmt und vergleicht, wie sich das aktuelle Bild im Vergleicht zu dem vorangegangenen verändert hat. Die einzelnen Bewegungen sollten allerdings relativ einfach gehalten sein, um der Hardware die Erkennung zu erleichtern und gleichzeitig dem Nutzer, der dieses System vermutlich zum ersten Mal bedient, ein einfaches Erlernen zu ermöglichen. Zur Zeit werden nur vier sehr einfache Zauberrunen erkannt - Erde (⊓), Wind (⊐), Feuer (⊔) und Regen (⊏).

Da die Geschichte - von der Einleitung im Museum abgesehen - ausschließlich im Freien spielt wird zur Positionierung der Spieler ein GPS-Empfänger verwendet. Dieser kann problemlos mittels Bluetooth mit den Eingabegeräten kommunizieren. Für weitere Versionen wird ebenfalls eine Übertragung der Position via UMTS in Betracht gezogen und zu diesem Zweck getestet.

Auf Grund dieser neuartigen Möglichkeiten, die REXplorer bietet, musste dieses Projekt neben den üblichen Problemen bei der Entwicklung weitere, ebenfalls neuartige Probleme lösen, wie dem Layout des Eingabegeräts, das auch ein unerfahrener Nutzer bedienen kann, der leichten Unterscheidbarkeit zwischen Gebäuden mit Funktionen im Spiel und normalen Bauwerken oder einer Gestenerkennung, die auch über leichte Abweichungen hinweg sieht. Zusätzlich mussten, wie in nahezu allen ortsbasierten Spielen, Ungenauigkeiten des GPS-Signals ausgeglichen werden. [WBB+06]

2.4.11. The Songs of North

The Songs of North ist ein Pervasive Multiplayergame, bei dem eine persistente Spielwelt erschaffen werden sollte. Der Spieler übernimmt dabei die Rolle eines Schamanen, der im Kontakt zu einer spirituellen Welt steht. Dazu steht ihm ein Totem-Tier zur Verfügung, das seinen Geist in dieser Welt verkörpert. Götter treten regelmäßig über das Tier in Kontakt mit dem Spieler und erteilen ihm Aufträge, um die Welt zu verbessern oder aber die persönliche Macht des Spielers zu steigern. Die Kombination aus der realen und der geistigen Welt ergibt die eigentliche Spielumgebung. In Abbildung 2.20 ist dieses Konzept zu erkennen. Der Schamane - oder genauer der Spieler selbst - kann sich ausschließlich in der Realität bewegen, während sein Geist die gleichen Pfade in der spirituellen Welt beschreitet. Interaktionen sind im Spiel über eine Trommel möglich auf der vordefinierte Befehle „eingegeben“ werden können. [Hypo4]

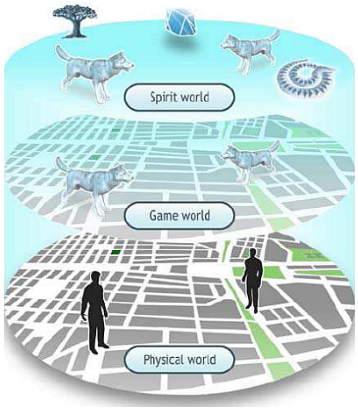

Abbildung 2.20.: Spielwelt von SoN (Quelle: [EEL+05])

Als Ausrüstung, um SoN spielen zu können, wird lediglich ein modernes Mobiltelefon mit GSM benötigt. Auf dem Display sieht der Spieler seine Trommel und kann diese mit den zehn Zifferntasten

bedienen. Dabei dient die Oberfläche der Trommel gleichzeitig als Karte auf der wichtige Orten der spirituellen Welt, die Position virtueller Gegenstände oder andere Mitspieler eingeblendet werden können. Die Lokalisierung des Spielers erfolgt durch die Bestimmung der Einwahlzelle des GSM Netzes. Sowohl die Positionsdaten, als auch die Eingabedaten werden als HTTP-Request an einen zentralen Spielserver gesendet, der diese auswertet.

Die puristische Grafik wird durch ein ausgeklügeltes Soundsystem ausgeglichen. Die Geräuschkulisse soll dem Spieler Ereignisse der spirituellen Welt vermitteln, da diese anders nicht wahrgenommen werden können. Beispielsweise werden in Kämpfen spezielle Töne für unterschiedliche Angriffe abgespielt oder es wird die aktuelle Stimmung der virtuellen Welt über dynamische Hintergrundmusik wiedergegeben. [EEL+05]

Bei diesem Projekt wurden drei Designaspekte von Anfang an besonders berücksichtigt. Zunächst sollte eine persistente Spielwelt zum Einsatz kommen. Dies ist sinnvoll, da die reale Welt ebenfalls nicht unterbrochen oder an bestimmten Punkten abgespeichert werden kann. Dennoch sollte es dem Spieler gestattet werden, Pausen im Spiel einzulegen. Dieses Problem wurde gelöst, indem man daraus ein Feature machte. Der Schamane kann, wann immer er will Kontakt mit seinem Geist aufnehmen und wieder abbrechen. So kann die virtuelle Welt weiter existieren aber nur eingeloggte Spieler können in dieser agieren. Allerdings ist es durchaus möglich ohne Verbindung zur spirituellen Welt einige Teilaufgaben von Missionen zu lösen, indem man sich zu Missionszielen bewegt und erst dann wieder Kontakt zum Server aufnimmt.

Ein weiterer Designaspekt beschäftigt sich mit dem Aufbau von sozialen Netzwerken unter den Spielern. Man entschied sich dazu, dass die Missionen auch alleine gelöst werden können. Dies ist aber - bedingt durch einen sehr hohen Schwierigkeitsgrad - nicht ratsam. Um eine Zusammenarbeit mit meist unbekannten Mitspielern zu ermöglichen wird ein Kommunikationsmittel benötigt. Zu diesem Zweck wurde ein sehr einfaches und günstiges Textmitteilungssystem implementiert. So können auch dauerhafte Gruppen gebildet werden, die von den Stärken der einzelnen Mitglieder profitieren können. Bestenfalls entstehen aus diesen Team vom Spiel losgelöste Communities. Die Privatsphäre eines einzelnen Spieler kann aber dennoch auf Wunsch gewahrt werden, da die Position seines Tier nur mit einer Ungenauigkeit weitergegeben wird. Erst wenn er sich freiwillig in der Realität zu erkennen gibt, können seine Mitspieler wissen wem das Tier gehört.

Schließlich sollten besondere Aspekte hinsichtlich der Ein- und Ausgabe bei Mobiltelefonen berücksichtigt werden. Die Trommel erleichtert dies beides. Zum einen ist die Grafik sehr schlicht, wodurch die benötigte Leistung gering bleibt und zum anderen werden für die Eingabe nur die ohnehin vorhandenen Zifferntasten benötigt. Die Bewegung erfolgt automatisch durch das GSM Netz, was ebenfalls bereits für das Telefon benötigt wird. [LHN+04]

Umfragen nach einer zweiwöchigen Testphase ergaben allerdings, dass der Spielspaß deutlich unter dem von herkömmlichen Spielen liegt, der Frustfaktor aber um ein vielfaches höher ist. Dies hing aber in erster Linie mit technischen Problemen bei der Lokalisierung der Spieler zusammen, da das ungenaue GSM Ortungssystem ein fehlerfreies Spiel verhinderte. Aber auch bei der Verbindung zum Spielserver kam es häufig zu Unterbrechungen. Da der Anteil der Offline Aktivitäten in SoN allerdings sehr gering ist, wollte sich aus diesem Grund ebenfalls kein Spielfluss einstellen. Grundsätzlich wurde aber durchaus großes Interesse an dieser Spielidee signalisiert. [EM05]

2.4.12. Uncle Roy All Around You

Ein weiteres Projekt von Blast Theory stellt Uncle Roy All Around You dar. Genau wie bei Can You See Me Now? (Abschnitt 2.4.5) wird versucht, die reale mit der virtuellen Welt zu verschmelzen,

indem professionelle Darsteller mit den Teilnehmern entsprechend den Regeln des Spiels interagieren. Auch bei Uncle Roy besteht das Spielfeld aus einem ausgewählten Bereich einer realen Stadt, vom dem ein virtuelles Modell angelegt wurde. Im Unterschied zu CYSMN gibt es allerdings neben den Onlinespielern hier auch Spieler, die sich ihren Weg durch die realen Straßen bahnen müssen. [BM08b]

Aus diesem Grund können auch zwei unterschiedliche Erfahrungen mit dem Spiel gemacht werden. Den Spielern, die sich durch die Straßen bewegen wird zu Beginn ein PDA mit GPS Empfänger ausgehändigt und sie erhalten die Aufgabe, innerhalb einer vorgegebenen Zeit Uncle Roy zu besuchen. Auf ihrem PDA erhalten sie Aufgaben, die gelöst werden müssen, um weitere Hinweise auf den Treffpunkt zu erhalten. Auf ihrer Tour durch die Stadt lässt das Spiel ihnen allerdings nicht nur sinnvolle Ratschläge zukommen, sondern auch irreführende. Hilfe können die Spieler allerdings von den Onlineteilnehmern erhalten, mit denen sie jederzeit Textnachrichten oder kurze Audiofiles austauschen können. Diese Spieler können sich an jedem beliebigen Punkt der Erde befinden und bewegen ihren Avatar, wie in Abbildung 2.21 zu sehen, durch ein Modell des Spielbereichs. Dabei können ihnen andere Online- und auch Offlinespieler begegnen. Die Idee hinter Uncle Roy sieht dabei vor, dass einige Hinweise nur von Onlinespielern, andere nur von Spielern vor Ort gelöst werden können. Um erfolgreich das Spiel beenden zu können, müssen daher beide Spielergruppen zusammenarbeiten, um beide Welten um den Kontext der jeweils anderen anzureichern. Erreichen die Spieler Uncle Roys Büro, so können sich die beiden Gruppen „treffen“, da der dafür vorgesehene Raum mit Webcams ausgestattet ist. [BSF+04]

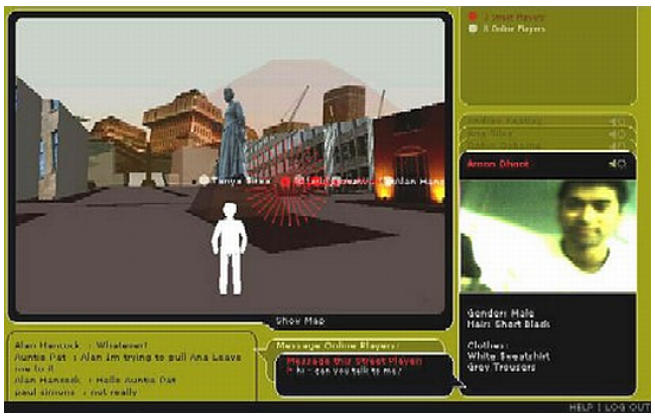

Abbildung 2.21.: Interface von Uncle Roy (Quelle: [Equo8])

Jedoch gibt es in Uncle Roy noch eine dritte Spielergruppe, die angestellten Darsteller. Diese können jederzeit - für die Spieler mehr oder weniger offensichtlich - in das Spiel eingreifen. Bereits beim Einstieg in das Spiel treffen die Offlinespieler auf einen Schauspieler, der neben der technischen Einführung auch direkt den Eindruck vermitteln soll, dass es sich nicht um ein Spiel handelt, sondern um eine reale Einladung. Jedoch ist die Verdichtung der Atmosphäre nicht die einzige Aufgabe der Darsteller. Sie werden auch zur Lösung einiger Aufgaben benötigt. Ein Hinweis könnte lauten, dass man einer schwarzhaarigen Frau folgen soll. Da das Spielgebiet nicht abgesperrt ist, können neben der eigentlich gesuchten Schauspielerin auch zufällig eine Passantin mit schwarzen Haaren dem Spieler begegnen und er muss sich entscheiden wem er folgt. Dadurch ergibt sich eine völlig neues Spielerlebnis, da kein Spieler wissen kann, wer zum Spiel gehört und wer nicht. [FAB+03]

Mit Uncle Roy werden vier Ziele verfolgt. Wie bei den meisten Pervasive Games sollte die virtuelle Welt möglichst fließend in die reale übergehen. Zumindest für die Offlinespieler gelingt dies sehr gut,

da es sich bei allen Hinweisen und alle Treffpunkten um real existierende Orte handelt, die besucht werden müssen. Das Spiel lebt aber auch davon, dass der Spieler immer im Ungewissen ist, ob er die Hinweise richtig deutet oder ob er der richtigen Fährte folgt, da die Spielwelt sehr groß ist und neben spielrelevanten auch völlig zufällige Inhalte einbezogen werden. Selbst an die Anfälligkeit des GPS-Signals wurde nicht nur gedacht, sondern diese Schwäche bewusst in das Spiel eingebaut. Kann die Position eines Spielers nicht bestimmt werden, so ist es seine Aufgabe diese möglichst genau an seine Onlinemitspieler weiter zu geben. In dieser Interaktion liegt auch der größte Reiz von Uncle Roy. Während die vorgegebenen Spielinhalte sehr übersichtlich sind - neben den Schauspielern und den Hinweisen wird nichts angeboten - so wird viel Wert auf den Aufbau von sozialen Kontakten gelegt. Wenn die Spieler sich gegenseitig unterstützen und regelmäßig mit Informationen versorgen, können alle davon profitieren. Nach einem erfolgreichen Abschluss des Spiels werden diese Bindungen noch weiter verstärkt, indem das System die Kontaktdaten von zwei Teilnehmern austauscht, die sich im Spiel bereits virtuell getroffen und geholfen haben. [BFD+04] Darin besteht auch der Sinn des Spiels, da kein Gewinner oder Verlierer im eigentlichen Sinn ermittelt wird, sondern nur der Erfolg der Gruppe im Vordergrund steht [BMB03].

Auch wenn das Spiel gute Ansätze hat und bei der Premiere bei den Teilnehmern - sowohl online (440 Teilnehme) als auch offline (272 Teilnehme) - sehr beliebt war [BSF+04], so fand seit Juni 2003 - bedingt durch den großen Aufwand für die Schauspieler und die benötigte Technik - kein weiteres Spiel mehr statt [BMB03].

2.4.13. Zusammenfassung

Die zwölf vorgestellten Pervasive Games stehen beispielhaft für eine Vielzahl vergleichbarer Projekte. Bei den Titeln handelt es sich zu meist um rein wissenschaftliche Projekte, da der Aufwand oder die zum Einsatz kommende Technik das Spiel zu teuer für den Massenmarkt machen würde. Allerdings gibt es auch einige wenige Ausnahmen wie Bot Fighters! 2.4.4. Hinsichtlich des Spielprinzip oder Genres lassen sich diese Spiele allerdings nicht einschränken. Es lassen sich sowohl strategische Endlosspiele als auch levelbasierte Action Shooter bei den untersuchten Titeln finden. Auffällig ist hingegen, dass der Großteil der Spiele sowohl einen Einzelspielermodus, als auch einen Mehrspielermodus anbieten. Oft wird dies als Stilmittel eingesetzt, um Technikausfällen entgegenwirken zu können, wenn beispielsweise die Position nicht bestimmt oder keine Verbindung zum Server aufgebaut werden kann. Die eingesetzte Grafik umfasst alle denkbaren Arten und wird maßgeblich durch das Genre und natürlich die verwendete Technik bedingt, genau wie die Möglichkeiten zur Spielunterbrechung.

In Tabelle 2.2 werden die vorgestellten Spiele in den ersten drei Kategorien (Spielprinzip, Unterscheidung von Einzelspielerspielen (SP) und Mehrspielerspielen (MP) und Setting) bewertet, während in Tabelle 2.3 neben den restlichen drei Bereichen (Genre, Grafik und Spielunterbrechung (BR)) auch die technischen Voraussetzungen, die die Möglichkeiten des N810 übersteigen oder einen Einsatz im Browser unmöglich machen, gefunden werden können. Bei den Möglichkeiten der Spielunterbrechung werden keine Spielunterbrechung (—), keine Spielunterbrechung mit Speicherung eines Highscores (HS), die Fortschrittsspeicherung (SP) und die Komplettspeicherung (voll) unterschieden.

2.5. Browserspiele

Bei Browserspielen handelt es sich um Programme, die in der Regel mit sehr vielen Mitspielern gespielt werden und vollständig im Browserfenster ablaufen können. Dadurch muss keine Software - neben dem Browser und eventuellen Plugins für den Ablauf von Skripten - installiert werden und

Spiel	Spielprinzip	Spielerzahl		Setting
		SP	MP	
AR² Hockey	Spiel auf Sätze		X	echter Airhockey Tisch
ARQuake	levelbasiert	X		fremden Dimensionen
Augmented Knight's Castle	n.v.	X	X	Mittelalter
Bot Fighters!	Endlosspiel		X	Zukunf
Can You See Me Now?	rundenbasiert		X	n.v
Epidemic Menace	Story	X	X	Gegenwart
Human Pacman	rundenbasiert		X	Spielautomat
MOPET	rundenbasiert	X		Personal Trainer
Pirates!	Endlosspiel	X	X	Piraterie
REXplorer	Story	X		Gegenwart
The Songs of North	Endlosspiel	X	X	alternative Realität
Uncle Roy All Around You	Story	X	X	Gegenwart

Tabelle 2.2.: Zusammenfassung der vorgestellten mobilen ortsbasierten Spiele I

eine Systemunabhängigkeit wird ermöglicht. Dies ist einer der Gründe, warum diese Spielesparte momentan boomt wie kaum eine andere. Im Gegenzug ermöglicht die große Verbreitung von privaten Breitband-Internetanschlüssen die Entwicklung von immer besseren Titeln. [Hö7]

Während die ersten Spiele beinahe ausschließlich aus rundenbasierten Strategie, Simulation oder Rollenspielen bestanden, bei denen entweder nur eine reinen Textausgabe oder eine Kombination aus Text mit sehr einfachen 2D-Grafiken zum Einsatz kommen, werden heute praktisch alle Genres bedient. Das Problem dabei ist, dass auf moderne Programmierschnittstellen, wie DirectX, um leistungsfähige Grafikkarten zu unterstützen, nicht zugegriffen werden können. [Wiko8e] Der Einsatz von clientseitigen Plugins für JavaScript oder Flash oder der serverseitigen Verwendung von AJAX in Kombination mit Sprachen wie PHP oder Java [Wiko8d], gestatten es dennoch ebenfalls actionreiche Titel mit einer starken Grafik-Engine, die in Echtzeit ablaufen, zu erstellen [Wiko8e]. Einzelspieler-Spiele laufen in erster Linie auf dem Client ab, während sich bei Mehrspieler-Spielen mit einer sehr großen Spielerzahl die Nutzung von serverseitigen Skripten empfiehlt [Wiko8d].

Führende Marktforschungsunternehmen gehen daher davon aus, dass dieser Markt auch weiterhin stark anwächst, obwohl die Zahl der kostenpflichtigen Titeln in diesem Bereich im Gegensatz zu klassischen Onlinespielen sehr gering ist. Meist sind nur Features wie beispielsweise besonders starke Gegenstände oder erweiterte Spielinformationen, zahlungswilligen Spielern vorbehalten. [Hö7] Dies sorgt dafür, dass sich häufig sehr große Communities bilden, die sich nicht nur im Spiel, sondern auch über andere internetnahe Medien, wie Foren oder Instant Messengern, austauschen [Wiko8e]. Die so entstehenden sozialen Netzwerke würden Pervasive Games ebenfalls zu Gute kommen.

Ein weiterer Vorteil, der für einen Browsereinsatz bei Pervasive Games spricht, ist, dass ein Browserspiel auf einem Internetserver abläuft. Somit ist eine Persistenz der Spielwelt gesichert, selbst wenn ein Spieler aussteigt. [Lubo8] Auch Endlosspiele sind auf diese Weise möglich und die bereist angesprochene Problematik der Spielunterbrechung bei herkömmlichen Spielen könnte umgangen werden.

Spiel	Genre	Grafik	BR	Technik
AR² Hockey	Simulation	3D	—	HMD, Spieltisch mit Markierungen
ARQuake	Action	3D	k.a.	HMD, hohe Rechenleistung, Eingabegerät, Passermarken
Augmented Knight's Castle	Sonstige	txt, 2D, 3D,	voll	große Königsritterburg, RFID, Eingabegerät
Bot Fighters!	Action	txt	voll	GSM
Can You See Me Now?	Sonstige	3D	HS	Darsteller
Epidemic Menace	Abenteuer	2D, 3D,	—	HMD, Eingabegerät
Human Pacman	Action	2D, 3D	HS	HMD, DRM sammelbare Gegenstände
MOPET	Sonstige	3D	HS	evtl. Sensoren
Pirates!	Action	2D	SP	—
REXplorer	Abenteuer	2D	—	Eingabegerät
The Songs of North	Abenteuer	2D	voll	GSM
Uncle Roy All Around You	Abenteuer	2D	—	Darsteller

Tabelle 2.3.: Zusammenfassung der vorgestellten mobilen ortsbasierten Spiele II

2.5.1. Cedysworld

Auch wenn es sich bei den klassischen Abenteuerspielen heute um ein im kommerziellen Sinn nahezu ausgestorbenes Genre handelt, so existiert für diese Spielart dennoch eine große Fanbasis. Dass von diesen Spielen außerdem keine revolutionäre Grafik oder Actionsequenzen in Echtzeit erwartet werden und die Steuerung meist mit nur einfachen Mausklicks auskommt, spricht für eine Umsetzung dieser Spiele als Browserspiel. [Bato2] Dennoch lassen sich vergleichsweise nur sehr wenige Vertreter dieses Genres im Internet finden.

Eine der wenigen Ausnahmen stellen einige Spiele der Seite Cedysworld, die von Mercedes-Benz als Lernseite für Kinder im Alter von 8 – 12 Jahren gedacht ist, dar. Die komplette Seite wurde in Flash entwickelt und ist der Zielgruppe entsprechend sehr bunt und comicartig gehalten. Bei den Spielen wurde konsequent darauf geachtet, dass neben der reinen Unterhaltung - und der mehr oder weniger unterschwelligen Werbung für die eigene Marke - der Wissenszuwachs nicht zu kurz kommt. [Mico4] Auch wenn die Seite registrierten Spielern einige Interaktionsmöglichkeiten mit anderen Mitgliedern bietet, wie beispielsweise einem Chat, so handelt es sich bei den Spielen um reine Einzelspielerspiele, wie es für klassische Adventures üblich ist.

Von den insgesamt achtundzwanzig Spielen, die auf der Seite angeboten werden fallen vier unter die Rubrik der Abenteuerspiele. Sieht man von dem Setting und der Hintergrundgeschichte ab, so unterscheiden sich Abenteuer im Museum, Abenteuer in Indien und Abenteuer in Südafrika kaum. Die eigene Spielfigur wird mit der Tastatur durch hauptsächlich statische Hintergrundbilder

gesteuert, in denen Gegenstände mit Interaktionsmöglichkeiten versteckt sind. Werden diese mit dem Mauszeiger angeklickt, so wird die der Situation und dem Gegenstand entsprechende Aktion ausgeführt, ohne dass der Spieler einen weiteren Einfluss darauf hat. Zum einen sorgt dies dafür, dass die meist sehr jungen Spieler nicht unnötig überfordert werden. Zum anderen ergibt sich dies aber auch durch den Einsatz von Flash, das keinen Rechtsklick abfangen kann und daher ein komplexere Auswahl von Aktionen auch die Ergonomie der Eingabe erheblich verschlechtern würde. Eingesammelte Objekte werden jederzeit im Inventar angezeigt und können ebenfalls mit einem einfachen Mausklick verwendet werden, sollte die aktuelle Situation dies erforderlich machen. Auch bei dem Abenteuer im Regenwald kommt eine ähnliche Technik zum Einsatz, allerdings wurde der Einsatz der Tastatur zur Navigation der eigenen Spielfigur ebenfalls durch Mausklicks ersetzt. Wird auf einen spielentscheidenden Gegenstand oder Ort geklickt, so begibt sich Cedy automatisch dorthin und führt gleichzeitig die verknüpfte Aktion aus.

Während man sich so von Szene zu Szene bewegt, erlebt man nicht nur eine Geschichte, sondern bekommt auch viele Hintergrundinformationen zu dem Themenbereich Automobilbau und Umweltschutz und allgemein zum Setting des Spiels. Diese können, sobald sie erhalten wurden, im (virtuellen) PDA auf Wunsch erneut abgerufen werden. Das vollständige Display wird in Abbildung 2.22 am Beispiel von Abenteuer im Museum dargestellt, wobei sich das Aussehen der restlichen Titel nur durch die Farbwahl oder die Anordnung der Elemente unterscheidet.

Abbildung 2.22.: Interface von Cedysworld (Quelle: vgl. [Daio8])

Neben dem reinen Sammeln von Gegenständen und Informationen wird der Spieler durch Rätseleinlagen, wie sie in keinem Adventure fehlen dürfen, gefordert. So gilt es eine zerrissene Nachricht wieder zusammen zu puzzeln oder die geheime Kombination eines Zahlenschlosses zu knacken. Insgesamt ist der Anteil der abwechslungsarmen Rätsel aber recht gering, wodurch auch die Menge der zu übertragenden Daten klein gehalten wird. Da die sehr einfachen Spiele aufgrund ihres Umfangs dennoch nicht auf einmal durchgespielt werden können, werden an bestimmten Stellen Codes angegeben, mit deren Hilfe bei einem Neustart erst ab diesem Punkt weitergespielt werden muss.

Auch wenn es die Spiele nicht erforderlich machen und es genügend andere Seiten im Internet gibt, auf denen Chaträume speziell für Kinder eingerichtet wurden, hat die Daimler AG anscheinend dennoch eine hohes Interesse daran, eine Community aufzubauen. Zu diesem Zweck kann sich der Spieler vor Spielbeginn in seinen Account einloggen und durch seine erbrachte Leistung virtuelle Taler erhalten. Diese können im seiteneigenen Shop direkt in andere, ebenfalls virtuelle Objekte eingetauscht werden. So wird eine Art Wettbewerb unter allen Clubmitgliedern erzeugt, da jeder die Besten Gegenstände aus dem Shop ergattern will. [Daio8]

2.5.2. Comunio

Bei Comunio handelt es sich um einen der größten deutschen Online Fußballmanagern. Dabei können eigene Fußball Ligen von mehreren Spieler erzeugt werden, in denen jeder Spieler die Leitung über ein eigenes Team übernimmt. Auch wenn es verschiedene Spielvarianten gibt, so ist das Grundprinzip in jedem Modus das gleiche: In einer Liga, im Sprachgebrauch des Spiels auch als Community bezeichnet, müssen die Spieler innerhalb eines fest vorgegebenen Budgets ein Team aus Fußballspielern aufbauen und eine Mannschaftsaufstellung und eine Taktik festlegen. Anhand des Erfolgs der Mannschaft werden nach jedem Spieltag Punkte und Geld unter den Spielern ausgeschüttet. Am Ende einer Saison wird der Gewinner über die erreichten Punkte ermittelt.

Zwei Aspekte, die bereits bei Pervasive Games auftauchten, werden auch bei Comunio berücksichtigt. Zum einen werden durch die Bildung einer Community soziale Kontakte geknüpft. Um Comunio überhaupt spielen zu können, muss man entweder eine neue Liga eröffnen oder einer Bereits bestehenden beitreten. In jedem Fall funktioniert der Spielablauf nur dann optimal, wenn sich mehrere Mannschaften, also mehrere Spieler, in einer Liga befinden. Jeder dieser Gruppen steht ein privates Forum zur Verfügung, in denen man sich neben spielbezogenen auch über alle möglichen privaten Themen austauschen kann. Aber auch für das Spiel selbst sollte mit man mit den Mitspielern interagiert werden. So können für Gegenspieler Gebote abgegeben oder Gebote von anderen Spielern an eigene Fußballer angenommen werden. Auch wenn dabei die Wertvermehrung im Vordergrund steht, so sollte berücksichtigt werden, dass ein Mitspieler, dessen Angebote man selbst mehrfach abgelehnt hat, wohl ein eigenes Angebot ebenfalls ausschlagen wird. Ein Spieler mit guten Beziehungen zu seinen Mitspielern hat demnach größere Chancen auf ein erfolgreiches Spiel, als ein typischer Einzelspieler.

Zum anderen fließt bei Comunio der Kontext der Spieler mit ein. Auch wenn die Ligen und Team frei erfunden sind, so sind die Fußballer Abbilder von echten Spielern, die in realen Ligen spielen. Die verteilten Punkte hängen allein von der Leistung dieses echten Spieler nach jeder Partie ab. Damit muss ein Spieler also auch Verletzungen, mögliche Transfers, Leistungskurven, usw. von der realen Liga im Auge behalten, um seine Team optimal zusammenstellen zu können. [como8] Neben mehreren verschiedenen Ligen und Pokalwettbewerben werden auch vollkommen andere Sportarten beispielsweise aus dem Wintersportbereich spielbar. Dieses Portfolio wird dabei schrittweise erweitert. [Wiko8f]

Dieses Spielprinzip ist sehr beliebt. So hatte Comunio 2006 bereits über 350000 Accounts, wobei die Anzahl stetig ansteigt. Dieser Erfolg lässt sich erklären, da nicht nur das Spiel durch reale Ergebnisse beeinflusst wird, sondern auch das Interesse an Spielen gesteigert oder erst geweckt wird, wenn man einen Spieler eines der beteiligten Teams besitzt. Auch der direkte Vergleich inklusive hitziger Diskussionen nach Spielende mit den Freunden wessen Spieler wohl die bessern Noten bekommen wird trägt zum Spielspaß erheblich bei.

Finanziert wird das theoretisch kostenlose Comunio zum einen über Werbung und zum andern über Spielgebühren für Profimodi, die Zusatzinhalte, wie mehr Informationen über das eigene Team oder weiter taktisch Möglichkeiten bereitstellen. [fabo6]

Wie man in Abbildung 2.23 sehen kann, kommt das Spiel mit einer Textoberfläche eingebettet in einer sehr einfachen 2D Oberfläche ohne Animationen aus und läuft nahezu ausschließlich serverseitig ab. Dabei kommen die Programmiersprachen PHP und JavaScript zum Einsatz. Mit JavaScript werden clientseitig die Eingaben des Spielers in einem normalen Webformular entgegengenommen. Die komplette Auswertung und die Spiellogik ist auf dem Server hinterlegt und ist daher für den Anwender völlig unsichtbar. Neben einem JavaScript-Interpreter werden keine besondere Plugins für den Browser benötigt. Die Nutzerdaten werden in einer MySQL-Datenbank hinterlegt. Der Spieler

Abbildung 2.23.: Interface von Comunio (Quelle: vgl. [como8])

wird beim Login zufällig auf einen von mehreren parallel betriebenen Servern umgeleitet, um eine optimale Lastverteilung zu ermöglichen. [Loso8]

2.5.3. Dancestar Online

Im Gegensatz zu den meisten Browserspielen wird Dancestar Online speziell für Konsolen, genauer gesagt der PS3, von Bigpoint entwickelt. Da die Ressourcen, die dem Browser der PS3 zur Verfügung stehen, aber sehr stark begrenzt sind, kann das Spiel auch auf anderen Systemen, auf denen ein Browser läuft, genutzt werden. Ziel des Projekts ist es, dem Spieler das Look and Feel eines normalen Offlinespiel zu bieten. So anspruchsvoll dieser Vorsatz ist, so einfach ist der Inhalt von Dancestar Online. Es handelt sich um einen schlichten Klon eines herkömmlichen Tanzmattenspiels. [Gamo8]

Tanzmattenspiele wie Dance-Dance-Revolution oder eben Dancestar Online gehören zur Familie der Bemani-Spiele. Dabei handelt es sich um Spiele, die von der Musikspielabteilung von Konami entwickelt wurden. Gemein haben alle dieser Titel, dass es sich um ein stark musikorientiertes Spielkonzept mit einer besonderen Eingabemöglichkeit handelt. Bei den genannten Tanzmattenspielen ist dies eine Sensorbodenplatte, auf der der Spieler vorgegebene Tanzsequenzen zum Takt der Musik nachtanzen muss. [Wiko8b]

Steht dem Spieler von Dancestar Online kein solches Eingabegerät zur Verfügung, so kann aber auch jedes andere Eingabemittel, das an das zum Einsatz kommende System angeschlossen werden kann, verwendet werden. Dann müssen beispielsweise die Pfeiltasten reaktionsschnell gedrückt werden, um einen möglichst hohen Punktestand zu erzielen. [Gamo8] Dieser Punktestand stellt zum jetzigen Zeitpunkt die einzige Möglichkeit für einen Art Mehrspielermodus dar, indem die Spieler nacheinander das selbe Lied „tanzen“ und ihre Punkte vergleichen. Eine Highscoreliste ist zwar vorgesehen, aber zum jetzigen Zeitpunkt noch nicht in der freigegebenen Version implementiert. [Bigo8]

Auch die Anzahl der im Spiel enthaltenen Level, beziehungsweise Lieder, ist noch auf zwei beschränkt. Hier soll eine Community einbezogen werden, die eigene MP3 Dateien hochladen können, an denen sie sich mit ihren Freunden messen können. Auf diese Weise ist eine Finanzierung des eigentlich kommerziell angelegten, aber augenblicklich kostenlosen Projekts denkbar, indem neue Inhalte nur zahlenden Nutzern zur Verfügung gestellt werden. [Stao8]

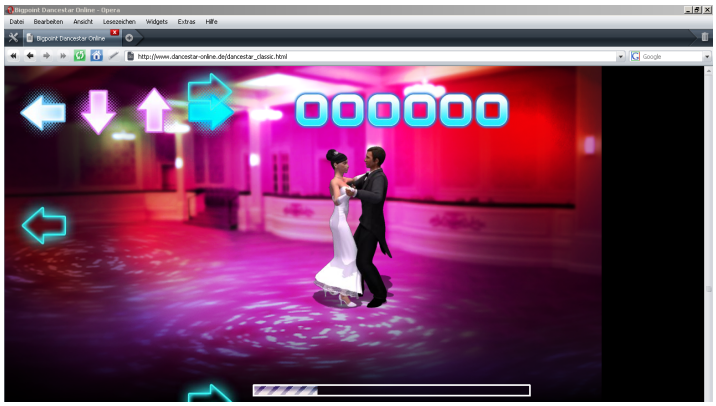

Abbildung 2.24.: Interface von Dancestar Online (Quelle: vgl. [Bigo8])

Wie man in Abbildung 2.24 sehen kann, erinnert nur noch die Browserleiste daran, dass das Spiel vollständig im Browser abläuft. Dies wird ermöglicht, da die Webseite von Dancestar Online lediglich eine Shockwave Flash Datei öffnet, die daraufhin ununterbrechbar abläuft. Daher wird neben einem Browser auch ein Flash Plugin benötigt. Sollen aber die neuen Ideen einbezogen werden, so werden Anpassungen an dem Interface, und damit an der eingesetzten Programmiertechnik, wie Webformulare zum Dateiupload, benötigt. [Bigo8]

2.5.4. Travian

Bei Travian handelt es sich um ein Aufbau Strategiespiel, das vollständig im Browser abläuft. Zur Zeit des Römischen Reichs muss der Spieler in der Rolle eines Stammesoberhaupt der Gallier, Germanen oder Römer versuchen ein kleines Dorf aufzubauen und neue Dörfer zu gründen. Dabei muss er sich zunächst um die Wirtschaft seines Volkes kümmern und die Produktion von Rohstoffen wie Holz, Lehm, Eisen oder Getreide anlaufen lassen. Dafür stehen ihm achtzehn Felder um sein Dorf zur Verfügung, die jeweils einen der Rohstoffe zur Verfügung stellen. Wie alle Aktionen läuft auch der Abbau in der persistenten Spielwelt in Echtzeit ab, egal ob der Spieler eingeloggt ist oder nicht. Steht die Wirtschaft, so kann das Hauptdorf weiter ausgebaut werden.

Wie Abbildung 2.25 zeigt, besteht auch das Dorf selbst aus einzelnen Feldern, die als Bauplatz für Gebäude benutzt werden können. Die Gebäude dienen zum einen dazu, die Wirtschaft weiter voran zu bringen, wie Rohstofflager oder Marktplätze und zum anderen zu militärischen Zwecken wie Kasernen oder Waffenschmieden. Wie bei Aufbauspielen üblich, kann dem Spieler zu Beginn des Spiels nur aus einigen wenigen Gebäuden wählen, während andere erst bei einer entsprechenden Dorfgröße freigeschaltet werden. Wurde das Dorf ausreichend erweitert, sollte man sich um die Expansion des Reichs kümmern, indem weitere Dörfer in der Welt gegründet werden.

Die Spielwelt besteht aus Planquadraten, die jeweils einen (potentielles) Dorf mit den angrenzenden Wirtschaftsfeldern repräsentiert. Allerdings ist die Welt bereits von den Mitspielern bevölkert. Mit diesen kann ein friedlicher Umgang gepflegt und Waren ausgetauscht werden oder durch den Einsatz von Soldaten versucht werden deren Dörfer zu erobern. Ziel ist es auf diese Weise das größte Volk, gemessen an der Einwohnerzahl, zu werden. [Trao8a]

Das Hauptaugenmerk der Strategien liegt dabei auf den Umgang mit den Mitspielern. So können aus losen Bündnissen auch feste Allianzen entstehen, die Vorteile für alle beteiligten Spieler bringen

Abbildung 2.25.: Dorf in Travian (Quelle: vgl. [Trao8a])

sollen. Diesen Allianzen werden Kommunikationsmöglichkeiten zur Verfügung gestellt, damit sie ihr Vorgehen miteinander absprechen und eine gemeinsame Taktik erarbeiten können. Auf diese Weise wird gleichzeitig ein soziales Netzwerk unter den Spieler aufgebaut, da auf den Kommunikationswegen auch private Nachrichten ausgetauscht werden können. Die Gründung von diesen Gruppen wird von Travian bewusst gefördert, indem bestimmte Ziele im Spiel nur in großen Gruppen erreicht werden können.

Von den über drei Millionen Spieler, die Travian weltweit begeistert, trifft man im Spiel nur eine Teilmenge, da die Spieler in unterschiedliche Welten eingeteilt werden, um die Server zu entlasten. So gibt es für jedes Land je nach Spielerzahl eine oder mehrere eigene Instanzen der Welt. Dieses große Interesse an dem Spiel liegt neben dem schnell erlernbarem und dennoch forderndem Spielprinzip auch daran, dass Travian kostenlos ist. Neben Werbebanner wird es durch erweiterte, kostenpflichtige Modi mit mehr Funktionen und der Möglichkeit Gold zu kaufen, um im Spiel Vorteile zu erhalten, finanziert. [Wiko8i]

Auch wenn Travian mit einer liebevollen Grafik im Comicstil umgesetzt wurde, so kommen die einzelnen 2D Darstellungen vollkommen ohne Animationen aus. Dafür kommt clientseitig eine Kombination aus HTML und CSS mit JavaScript Formularen und weiteren JavaScript Ereignissen zum Einsatz, während die auf dem Server hinterlegten Seiten in PHP geschrieben wurden und auf einer MySQL-Datenbank aufsetzen. Die Kommunikation zwischen Client und Server wird mit dem AJAX-Konzept realisiert. Für die einzelnen Spielwelten stehen jeweils mehrere Apache und Ngix Webserver zur Verfügung, die je nach Last dynamisch zugeschalten werden. [Aggo8]

2.5.5. Travianer

Travianer stellt ein Spin-Off des beliebten Browserspiels Travian (siehe Abschnitt 2.5.4) dar, das ebenfalls von der Travian Games GmbH entwickelt wurde. Während das Setting und der Grafikstiel vollständig vom „Original“ übernommen wurde, ist die Spiellogik aber eine vollkommen andere. So ist Travianer ein Rollenspiel, in dem der Spieler einen Einwohner eines Dorfes, wie man es aus Travian bereits kennt, kontrolliert. Im Leben des Travianers spielen die bekannten vier Rohstoffe Holz, Lehm, Eisen und Getreide eine wichtige Rolle. So ist es die erste Aufgabe diese in den Feldern um das Dorf abzubauen. Im Gegensatz zu Travian sind die Produktionsmöglichkeiten des Spielers an dieser Stelle noch nicht abgeschlossen. Aus den Grundrohstoffen können weitere gewonnen werden.

So kann Getreide in einer Mühle zu Mehl gemahlen werden, das wiederum in einer Bäckerei zu Brot wird. Dabei gewinnt man für jede Aktion rollenspieltypisch Erfahrungspunkte und kann spezielle Berufe wie Bäcker erlernen, um die Verarbeitung Mehl zu Brot erfolgreicher durchführen zu können. Der Rohstoffabbau dauert aber nicht nur eine bestimmte Zeit, sondern es werden auch Berufspunkte verbraucht. Diese Berufspunkte werden zwar jeden Tag wieder automatisch aufgeladen, dennoch bedeutet diese Einschränkung, dass die tägliche maximale Rohstoffgewinnung begrenzt ist. Weitere Rohstoffe können auf einem Markt erworben werden.

Erst wenn ausreichend Rohstoffe gewonnen wurden, darf der Spieler das Wirtschaftstutorial verlassen und das Dorf - und damit das eigentliche Spiel - betreten. Neben den angesprochenen Märkten und höheren Produktionsstätten findet man hier Aufgaben, deren Erfüllung neben materiellen Belohnungen auch Erfahrungspunkte bringen, die den eigenen Charakter verbessern. Eine der Belohnungen ist ein eigenes Haus, das nach eigenen Wünschen eingerichtet werden kann - einen entsprechend gut gefüllten Geldbeutel vorausgesetzt. Neben der Charakterentwicklung soll dieser Ausbau für die Langzeitmotivation bei den Spielern sorgen, da man in Travianer nicht nur auf computergesteuerte Charaktere trifft, sondern auch auf viele menschliche Mitspieler, die durch die eigene Leistung beeindruckt werden sollen.

Dabei wurden dem Spieler viele weitere Möglichkeiten zum direkten Wettkampf mit seinen Mitspielern gegeben. In einer Arena kann man gegen den Charakter eines Mitstreiters antreten. Dabei wird abwechselnd rundenweise angegriffen und verteidigt. Neben der gewählten Taktik hängt der Kampfausgang von den Charakterwerten, also der Erfahrung aus den Aufgaben, ab. Weniger feindselig können auch Minispiele in einer Taverne gespielt werden. Dabei stehen bekannte Glücksspiele wie Blackjack, Würfeln aber auch Skat zur Auswahl.

Auch wenn diese Wettkämpfe nicht spielentscheidend sind, so kann damit neben dem Spielspaß auch sogenannte Sozialpunkte gewonnen werden, mit denen das eigene Haus ausgebaut werden kann. Diese Sozialpunkte stellen einer von mehreren Versuchen der Travian Games GmbH dar, den Spieler dazu zu bringen, Travianer nicht als reines Einspielerspiel - als das es theoretisch gespielt werden kann - sondern als quasi echtes, massives Mehrspielerrollenspiel anzusehen. Auch an anderen Stellen wird versucht den Aufbau von sozialen Bindungen voranzutreiben. So ist es beispielsweise die Aufgabe des Spielers einen Freund im Spiel zu finden. Dazu steht ihm jederzeit ein Chatfenster zur Verfügung, in dem alle aktiven User kontaktiert werden können. Hat man einige Freunde gefunden, so können diese sich fest zu einer Gilde zusammenschließen. Eine Gilde kann das soziale Gefüge weiter stärken, da innerhalb einer Gilde Waren gespendet werden können und so sozial schwachen Mitgliedern ausgeholfen werden kann. Auch die Möglichkeit Gildenkriege gegen andere Teams auszutragen kann die Mitglieder stärker zusammenschweißen.

Auch das Finanzierungsmodell wurde von Travian übernommen. Das normale Spiel kann völlig ohne jegliche Kosten gespielt werden ohne auf größere Einschränkungen zu stoßen. Allerdings kann eine Mitgliedschaft im sogenannten Goldclubs erworben werden, die einige Vorteile gegenüber den restlichen Spielern schaffen, wie beispielsweise mehr Berufspunkte pro Tag oder ein größeres Lager.

Wie man in Abbildung 2.26 sehen kann, erinnert die Grafik sehr stark an Travian, aber im Gegensatz zu den statischen 2D Grafiken, findet man sich hier in einer animierten Welt wieder. Gesteuert wird die Spielfigur wie im einem klassischen Point-And-Click Spiel. Klickt man auf ein besonderes Feld, wie beispielsweise auf die Bäckerei, so wird automatisch die entsprechende Aktion (Brot backen) ausgeführt.

Trotz dieser schönen Oberfläche kommt das Spiel vollkommen ohne besondere Plug-ins oder Erweiterungen für den Browser aus. Dies ist möglich, da auf AJAX-Technologie gesetzt wird. [Trao8b] So wurde bei der Entwicklung Xajax verwendet, das ein Toolkit für Ajax-Anwendungen auf Basis von PHP darstellt.

Abbildung 2.26.: Dorf in Travianer (Quelle: vgl. [Trao8b])

2.5.6. Yetisports

Bei Yetisports handelt es sich um eine Sammlung von kleinen Geschicklichkeits- und Actionspielen. Diese Flashspiele wurden von Chris Hilgert erfunden und von der Edelweiss Medienwerkstatt produziert. Im Gegensatz zu den restlichen aufgeführten Browserspielen verzichtet diese Reihe auf eine Hintergrundgeschichte oder Tiefgang. Einzige Gemeinsamkeit der Titel ist die namensgebende Hauptfigur - ein Yeti - der sich in verschiedenen sportlichen Disziplinen beweisen muss.

Ziel der Minispiele ist es eine möglichst gute Leistung zu erbringen, die anschließend mit Punkten bewertet wird. Aus diesen Punkten wird in Echtzeit eine Highscore Tabelle erstellt, so dass man sich nach jeder Runde direkt mit tausenden von anderen Spielern auf der ganzen Welt messen kann. Daraus resultiert die immense Beliebtheit der Reihe, da die Spieler so immer wieder angetrieben werden noch ein meist nur wenige Sekunden langes Spiel zu starten. [Wiko8j]

Auch die Spiele an sich sind sehr einfach gehalten. Oft besteht die Steuerung des kompletten Spiels aus einem einzigen, reaktionsschnellen Mausklick. Die Grafik ist zwar comicartig und detailverliebt gehalten, aber dennoch verglichen mit anderen aktuellen Flashtiteln sehr einfach, wie man anhand der vier Beispielen aus Abbildung 2.27 erkennen kann. [Hilo7]

Finanziert wird das kostenlose Spiel durch Merchandise-Produkten rund um die Yetiwelt und durch kostenpflichtige Portierungen für Mobiltelefone. [Wiko8j] Um die Verkaufszahlen dafür hoch zu halten wird versucht, eine aktive Community an das Spiel zu binden. Dazu wurde eine kostenlose Pflichtregistrierung eingeführt, um den ehemals anonymen Spielern ein einzigartiges Profil zu verleihen. Als Anreiz dafür werden neben neuen Disziplinen auch spezielle Zweispielermodi veröffentlicht, in denen sich die Spieler direkt messen können und nicht nur über ihrer Punktezahl. Auch Yetisports-Teams können auf der Seite gegründet werden, die in der Gruppe versuchen können immer bessere Leistungen zu erbringen.

Bei der Entwicklung der Spiele kam lediglich Flash zu Einsatz. Daher wird neben einem Browser ein entsprechendes Plugin benötigt, das die übermittelte SWF Datei clientseitig ausführen kann. Obwohl der zu übertragende Code pro Spiel nicht sehr groß ist, so ergibt sich bei 800.000 Zugriffen pro Tag einen monatlichen Traffic von bis zu zehn Terabyte. Dennoch kommt nur ein einziger x346 Server von IBM zum Einsatz. [Hilo7]

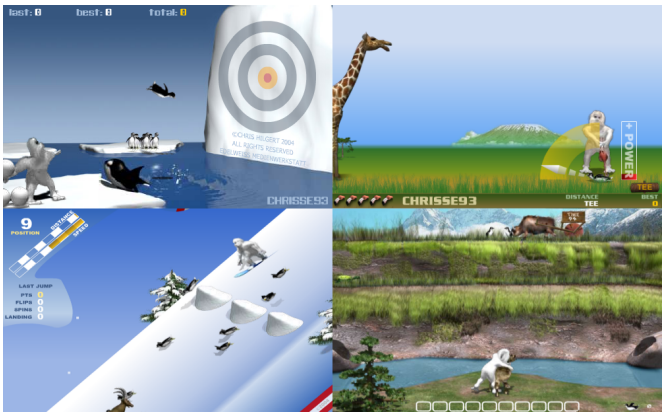

Abbildung 2.27.: Screenshots aus vier Yetisportstiteln (Quelle: vgl. [Hilo7])

2.5.7. Zusammenfassung

Auch wenn die Mehrheit der heute angebotenen Browserspiele im Bereich der Strategiespiele und Aufbausimulationen liegt [Wiko8e], so zeigen die oben genannten Beispiele, dass auch nahezu alle andern Spielgenres bedient werden. Gerade durch den kombinierten Einsatz neuer Technologien im Umfeld der Webprogrammierung wie AJAX und Flash lassen sich neuartige Spielarten im Browser umsetzen, die sich von herkömmlichen Spielen kaum noch unterscheiden lassen [Wiko8e].

Ein Beispiel für diese neue Generation der Browserspiele stellt eindeutig QuakeLive dar, das von id Software entwickelt wird und sich momentan in einer Beta Phase befindet. Bei diesem Projekt handelt es sich um eine Weiterentwicklung des beliebten eSports-Titels „Quake III Arena“, die ausschließlich für Webbrowser umsetzt wird, da darin für id die Zukunft der eSports-Spiel sieht. [Wiko8h] Leider sind die offiziellen Informationen zu diesem Titel sehr spärlich und auch auf mehrfache Anfragen wurden mir seitens der Entwickler keine weiteren Angaben gegeben. Dieses Spiel beweist in Kombination mit dem bereits genannten ARQuake aus Abschnitt 2.4.2, dass - einen entsprechenden Entwicklungsaufwand vorausgesetzt - sich wohl die meisten Pervasive Games für einen Browser umsetzen lassen würden.

Auch bei den vorhandenen Spielprinzipien lassen sich sehr viele Varianten finden. Von Endlosspielen, die in persistenten Spielwelten ablaufen und auch ohne eingeloggte Spieler auf dem Server weiterlaufen, über Spielen, bei denen dem Nutzer eine Geschichte erzählt wird, die durch seine Aktionen beeinflusst und vorangetrieben wird, bis hin zu sehr kurzen actionreichen Titeln, die nur wenige Minuten oder sogar nur einige Sekunden dauern und zu häufigen Wiederholungen motivieren sollen, ist nahezu alles geboten. Die Art des Spielprinzips hängt dabei stark vom Genre des Spiels ab, da beispielsweise eine Simulation erst durch eine ständig veränderliche Umwelt interessant wird und daher über einen sehr langen Zeitraum gespielt werden sollte. Mit dem Spielprinzip ergibt sich auch sehr schnell das System zur Unterbrechung des Spiels. Während bei sehr kurzen Titeln eine Sicherung des Highscores vollkommen ausreicht, müssen je komplexer das Spiel ist auch mehr Informationen zum Spielstand gespeichert werden.

Als Programmiersprachen kommen meist Kombinationen aus JavaScript für den Client und PHP auf dem Server zum Einsatz. Die Kommunikation der beiden Seiten wird mittels AJAX geregelt. Damit können sehr einfach textbasierte Spiele, aber auch statische 2D Titel erzeugt werden. Werden Animationen im Spiel benötigt, so kommt bei den Untersuchten Programmen zusätzlich - bei einfachen Titeln auch ausschließlich - Flash zum Einsatz.

Des Weiteren fällt bei allen betrachteten Titel auf, dass diese entweder einen Mehrspielermodus anbieten, ein solcher in Planung ist oder zumindest um das Spiel eine aktive Community aufgebaut werden soll. Daher scheint es unabkömmlich zu sein, in Browserspielen eine entsprechende Infrastruktur bereitzustellen, damit die Mitglieder auch außerhalb der Spielwelt in Kontakt bleiben können.

Das Setting ist auch bei Browserspielen zunächst vollkommen unbedeutend, da jedes Spielprinzip und jedes Genre in einer beliebigen Umgebung realisiert werden kann. Dennoch sollte die Spielumgebung passend gewählt werden, damit sich ein stimmiges Gesamtbild ergibt. Auf diese Weise können, wie beispielsweise in „Cedysworld“, auch didaktische Aufgaben durch das Spiel wahrgenommen werden oder wie im Fall von „Comunio“ ein Bezug zu einem realen Kontext geschaffen werden. Allerdings können auch Titel wie „Yetisports“ viele Spieler überzeugen, auch wenn die Spiele nicht nur in keinem Zusammenhang zum gegebenen Setting stehen, sondern auch die Aufgaben, die dem Spieler gestellt werden vollkommen sinnlos erscheinen.

Spiel	**Spielprinzip**	**Spielerzahl**		**Setting**
		SP	**MP**	
Cedysworld	Story	X		Gegenwart
Comunio	rundenbasiert	X	X	reale Fußballligen
Dancestar Online	liedbasiert	X	(X)	Gegenwart
Travian	rundenbasiert	X	X	Römisches Reich
Travianer	Point-And-Click	X	X	Römisches Reich
Yetisports	Spielesammlung	X	(X)	Minispiele

Tabelle 2.4.: Zusammenfassung der vorgestellten Browserspiele I

In Tabelle 2.4 werden das Spielprinzip, die Unterscheidung zwischen Einzelspielerspielen (SP) und Mehrspielerspielen (MP) und das Setting für die untersuchten Titel in stark komprimierter Form wiedergegeben, während sich in Tabelle 2.5 die fehlenden Angeben zu dem Genre, der Art der Grafik - textbasiert (txt), 2D oder 3D - der Möglichkeit zur Spielunterbrechung - keine Spielunterbrechung (—), Speicherung eines Highscores (HS), Fortschrittsspeicherung (SP) und Komplettspeicherung (voll) - und die verwendete Programmiersprache zu finden sind.

Spiel	**Genre**	**Grafik**	**BR**	**Sprache**
Cedysworld	Abenteuer	2D	SP	Flash
Comunio	Simulation	txt	voll	PHP
		2D		JavaScript
Dancestar Online	Sonstiges	3D	(HS)	Shockwave Flash
Travian	Strategie	txt	voll	PHP
		2D		JavaScript
Travianer	Rollenspiel	2D	voll	AJAX
Yetisports	Actionspiel	2D	HS	Flash

Tabelle 2.5.: Zusammenfassung der vorgestellten Browserspiele II

Kapitel 3

Eingesetzte Technik

In diesem Kapitel sollen die wichtigsten Techniken, die für das Framework und damit den Prototyp entscheidend sind, kurz beleuchtet werden. Dabei handelt es sich sowohl um Hardware-Aspekte, die der Drahtlosen Kommunikation oder der Ortung eines Spielers dienen, als auch um softwareseitige Hilfsmittel, die bei der Programmierung zum Einsatz kommen. Auch wenn es sich hierbei um Key Enabler für mobile ortsbasierte Browserspiele handelt, so beschränkt sich die Beschreibung dieser Techniken auf das Nötigste, da davon ausgegangen wird, dass der kundige Leser mit diesen bereits vertraut ist.

3.1. Drahtlose Kommunikation

Die Verbindung von mehreren Computern zu einem Netzwerk wurde früher mit Kabel, die die jeweiligen Geräte mit einem zentralen Verteilerknoten verbanden, realisiert. Für stationäre Computer ist diese Technik auch ausreichend. Mit zunehmender Verbreitung von mobilen Geräten, wie Notebooks, PDAs, usw. und dem damit verbundenen Wunsch auch mit diesen Netzwerke betreiben zu können, kam jedoch ein Bedarf an einem Ersatz für die unflexiblen Verbindungskabel auf. Die einfache Lösung stellt eine Kombination aus Sender und Empfänger dar, die mittels Funkübertragung den Datenverkehr von einem Endgerät bis zu einem Verteiler übernehmen. [HW03]

Zwei dieser Funktechniken kommen für das in dieser Arbeit zum Einsatz kommende Gerät in Frage und sollten daher etwas genauer betrachtet und verglichen werden.

3.1.1. Wireless LAN IEEE 802.11

Wird im Folgenden über WLAN gesprochen, so ist die von der IEEE unter der Bezeichnung 802.11 standardisierte Technik zur drahtlosen Kommunikation gemeint und ist nicht als allgemeiner Sammelbegriff für kabellose Netzwerke zu verstehen. Die Kommunikation verläuft sehr ähnlich wie bei dem herkömmlichen Ethernet und es werden die selben Schnittstellen zu der Sicherungsschicht des ISO-OSI-Modells angeboten, wodurch die beiden Techniken beliebig ausgetauscht werden können. [Nap06] Je nach verwendetem Übertragungsstandard werden (theoretisch) bis zu 600 MBit/s an Daten übermittelt, wobei dies auch von der Signalqualität, der Codierung, der Entfernung Sender – Empfänger und der Anzahl der Hindernisse zwischen diesen, beeinflusst wird. Auch wenn es verschiedene Betriebsmodi gibt - wie beispielsweise der direkten „Ad-Hoc“ Verbindung zwischen zwei Geräten - so wird für diese Arbeit nur die Verbindung zu einem zentralen Access Point, der die

Verbindung aller in Reichweite befindlichen Geräte mit einem kabelgebundenen Netzwerk herstellt, von Interesse sein. Dabei darf die Entfernung den theoretisch bestimmten Wert von 300 Meter keinesfalls übersteigen. Je nachdem wie die Umgebung beschaffen ist, muss dieser Wert stark nach unten korrigiert werden. In Gebäuden sinkt er sogar auf unter 10 Meter. Da auch sensible Daten übertragen werden und die Signale jederzeit von Dritten abgehört werden könnte, bietet der Standard selbst bereits mehrere Möglichkeiten zur Verschlüsselung an, von denen das WPA2-Verfahren (siehe [Wi-07]) als sicherstes gilt. Gerade für den Einsatz auf mobilen Geräten ist ein sogenannter „Power Save Mode“ vorgesehen, der die Sende- und Empfangseinheit deaktiviert, wenn keine Datenpakete verschickt werden müssen, um so die Batterie zu schonen. [Sau08]

3.1.2. Bluetooth

Da trotz Energiesparmodus WLAN gerade für mobile Geräte mit einer eingeschränkten Akkuleistung zu viel Strom verbraucht wurde 1994 von der Firma Ericsson Mobile Communications nach einer anderen Möglichkeit gesucht, um solche Apparaturen dennoch drahtlos mit ihrer Peripherie verbinden zu können [Nap06]. Für diesen Anwendungsfall wurde eine geringe Reichweite und eine eingeschränkte Übertragungsrate in Kauf genommen. Das Ergebnis ist Bluetooth, das einen Datentransfer über eine maximale Strecke von 100 Meter mit lediglich 723 kbit/s ermöglicht. Im Gegensatz zu WLAN, bei dem ein Netzwerk um einen zentralen Access Point herum aufgebaut wird, besteht ein Bluetooth aus zunächst gleichgestellten Geräten, die untereinander einen sogenannten „Master“ ermitteln, der die Kommunikation regelt. Ein Betrieb eines „Network Access Points“, der eine Verbindung mit einem Kabelnetzwerk herstellen kann ist zwar ebenfalls möglich, jedoch sollte für diesen Zweck WLAN bevorzugt werden. [HW03] Auch bei Bluetooth ist eine Datenverschlüsselung mit einem 128 Bit Schlüssel möglich. Um auch unter widrigen Bedingungen sicher senden und empfangen zu können, wird zur Übertragung keine feste Frequenz gewählt, sondern der Master bestimmt abhängig von den Datenpaketen eine eigene. So können Überlagerungen mit andern Bluetooth-Netzen oder WLAN-Sendern minimiert werden. Die größte Stärke liegt allerdings in der geringen Leistungsaufnahme. Dies wird in erster Linie durch ein dreistufiges Herunterfahren der Bloothooth-Komponenten während einer Übertragungspause erreicht. Je nach Anwendung kann somit der Energiebedarf flexibel angepasst werden. [Sau08]

3.1.3. Zusammenfassung

Die Vor- und Nachteile dieser beiden Techniken sollten in Tabelle 3.1 genauer betrachtet werden.

Fakt	**WLAN**	**Bluetooth**
Flexibilität	-	+
Geschwindigkeit	+	-
Leistungsaufnahme	-	+
Nutzerdichte	+	-
Reichweite	+	-
Signalrobustheit	-	+

Tabelle 3.1.: Vergleich von WLAN mit Bluetooth

Diese Gegenüberstellung soll allerdings keinen „Gewinner“ präsentieren, da gerade durch das Zusammenspiel unter Ausnutzungen der jeweiligen Stärken, das beste Ergebnis erzielt werden kann.

Diese großen Unterschiede der beide Techniken ergeben sich aus der unterschiedlichen Ausrichtung der beiden Systeme. [HW03] Für die stabile und schnelle Vernetzung von Computern sollte WLAN zum Einsatz kommen, während Bluetooth eine komfortable und akkuschonende Verbindung zu peripheren Geräten ermöglicht. Ist die WLAN-Signalstärke zu gering, so kann allerdings versucht werden auf ein Bluetooth-Netzwerk zurück zu greifen, da dieses weniger anfällig auf Signalstörungen reagiert. [Sau08]

3.2. Positionsbestimmung

Ortsbasierte Anwendungen und Pervasive Games, wie sie bereits in den Kapiteln 2.3 und 2.4 vorgestellt wurden, benötigen in der Regel relativ genaue Positionsdaten, an denen sich der Nutzer, beziehungsweise das Gerät auf dem sie ablaufen aufhält, damit sie auf Points Of Interest eingehen können. Dabei kann es sich, anwendungsabhängig, um reelle Orte, wie Hotels oder Tankstellen, aber auch zufällige Orte, beispielsweise der Lageplatz eines virtuellen Gegenstandes bei einer Schatzsuche, handeln. Diese Ortsinformationen sollen dabei automatisch erfasst und verarbeitet werden können. Dafür existieren mehrere Systeme, wobei GPS (und dessen Weiterentwicklungen) für pervasive Anwendungen die entscheidendste Rolle einnimmt. [AKY07] Daher soll in der Folge nur auf GPS eingegangen werden, während für weitere, teilweise ergänzende, Möglichkeiten der Positionsbestimmung, wie Timing Advance oder Assisted GPS, bei denen zusätzlich die Infrastruktur von drahtlosen Netzwerken ausgenutzt werden, auf die Literatur[1] verwiesen wird.

3.2.1. GPS

Das „Global Positioning System“ ist ein satellitengestütztes Ortungs- und Zeitbestimmungssystem. Mit Hilfe von 24 Satelliten, von denen zu jeder Zeit an jedem Punkt der Erde mindestens fünf sichtbar sind, soll eine weltweite Verfügbarkeit gewährleistet werden. Für eine exakte Positionsbestimmung müssen mindestens vier unterschiedliche Signale gleichzeitig störungsfrei empfangbar sein. Da das System ursprünglich rein militärischen Zwecken diente, wurden daran sehr hohe Anforderungen gestellt:

- Die Abweichung der ermittelten Position zu der realen darf nicht mehr als 10 – 30 Meter betragen.
- Jede Anwendung muss die Daten in Echtzeit erhalten können.
- Die Positionsbestimmung muss jederzeit sehr schnell erfolgen können.
- Jeder Punkt der Erde muss abgedeckt werden.
- Das System muss zu einem gewissen Grad fehlertolerant sein, um auch bei schlechten Wetterbedingungen zu funktionieren.
- Die Empfänger dürfen nicht unnötig groß sein.

Allerdings strahlen die Satelliten zwei unterschiedliche Informationen aus. Neben einem verschlüsselten Code, der nur militärischen Einrichtungen zugänglich ist - der sogenannte Precise Code - existiert auch ein freie zugänglicher Teil, der als Coarse / Acquisition Code bezeichnet wird. [Tsu00] Dieser frei zugängliche Teil des Signals kann allerdings einigen Einschränkungen, hinsichtlich der Stabilität und der Genauigkeit, unterliegen. Dennoch eignet es sich sehr gut für den Einsatz in vielen Anwendungsbereichen, wie beispielweise für Navigationsgeräte oder ortsbasierte Spiele [Dan97].

[1] siehe [Fri05], [Rot05] oder [ZGL03]

Allerdings ist eine Positionsbestimmung mittels GPS innerhalb von Gebäuden oder sehr dicht besiedelten Bereichen nicht möglich, da die Signale zu stark abgelenkt werden. Sollen ortsbasierte Anwendungen in diesen Bereichen funktionieren, so muss das GPS um andere Systeme erweitert werden, die - möglichst nahtlos -, sobald kein Satellitensignal mehr empfangen werden kann, die Lokalisierung zeitweise übernehmen können. [CL08]

3.3. Webprogrammierung

Während man früher unter Webprogrammierung lediglich das Erstellen einer statischen HTML-Seite, auf denen sich Informationen in einer strukturierten Form problemlos präsentieren lassen, verstand, so reicht diese Technik nicht mehr aus, wenn die Seiten interaktiv sein sollen, wie es unter anderem auch für Browserspiele unabdingbar ist. Aus diesem Grund wurden mehrere Scriptsprachen entworfen, die sich direkt in den HTML-Code einbinden lassen und entweder serverseitig (beispielsweise als JSP) oder clientseitig (wie JavaScript) zur Laufzeit ausgeführt werden.

Damit lassen sich zwar bereits einfache Anwendungen implementieren, allerdings bleibt das Problem, dass auch bei kleinsten Änderungen - beispielsweise wenn eine Spielfigur um ein Feld weiter bewegt wird - die vollständige Webseite neu aufgebaut werden muss. Gerade bei umfangreichen Seiten ist dieses Verhalten nicht akzeptabel. AJAX, eine Kombination mehrerer Technologien, schafft hierbei Abhilfe. [HPS06]

Da JavaScript nicht für die Realisierung großer Softwareprojekte konzipiert wurde, erweist sich die Verwaltung und Pflege des Quellcodes als nahezu unmöglich. Unter anderem wurde aus diesem Grund 2006 das Google Web Toolkit (kurz gwt) entwickelt. Damit kann der clientseitige Anteil des AJAX Codes vollständig in Java verfasst und anschließend in JavaScript umwandelt werden. Dank einer aktiven Community stehen mittlerweile - neben den bereits im Framework enthaltenen - sehr viele weiter Widgets zur Verfügung. [HT07] Da das gwt auch in dieser Arbeit bei der Entwicklung des Frameworks und des Prototyps zum Einsatz kam, wird an dieser Stelle auch darauf näher eingegangen, bevor dieser Abschnitt mit der Vorstellung von Adobe Flash, einem vollkommen anderen Ansatz der Webprogrammierung, abgeschlossen wird.

3.3.1. AJAX

Auch wenn AJAX häufig als Enabler für das Web 2.0 und damit für neue interaktive Webseiten angesehen wird, so ist die dabei verwendete Technik nichts Neues im Bereich der Webprogrammierung. Viel mehr handelt es sich um ein Konglomerat von bestehenden Techniken, die, geleitet von einer gemeinsamen Idee, neuartig eingesetzt werden. Um AJAX-Anwendungen erstellen zu können, sollte man daher auch einen Überblick über die eingesetzten Schlüsseltechnologien bekommen. Im Folgenden wird zunächst erklärt was AJAX bedeutet, bevor dessen einzelnen Bestanteile vorgestellt werden.

Bei herkömmlichen Webanwendungen muss der Nutzer mit einer Aktion bewusst einen HTTP-Transfer auslösen, um Zugriff auf neue Inhalte zu erhalten. Beispielsweise kann dies mit statischem HTTP - durch Hyperlinks - implementiert werden oder durch das Absenden von Formulardaten. In jedem Fall werden nur dann Daten an den Server geschickt oder neue angefordert, wenn eine entsprechende Anforderung vom Nutzer ausgeht. Eine solche Beschränkung ist aber keinesfalls zwingend erforderlich. Es ist ohne Weiteres vorstellbar (und teilweise wünschenswert), dass die Eingabe eines Nutzers in ein Webformular bereits vor dem Absenden überwacht, und, gegebenenfalls, mit Hilfestellungen, wie beispielsweise einer Wortergänzungen oder einer Rechtschreibprüfung,

unterstützt wird. Clientseitig wird dabei lediglich JavaScript benötigt (A**JA**X), das nahezu jeder Browser unterstützt.

Dies erbringt aber zunächst keinen großen Vorteil. Stellt man sich das oben genannte Beispiel vor, wobei alle Hilfestellungen von einem Server bezogen werden müssen, so würde, ohne den Einsatz von AJAX, die Webseite nach jeder Eingabe vollständig neu aufgebaut werden. Dies würde aber nicht nur einen unnötigen Traffic erzeugen, da nur eine (stark) eingeschränkte Teilmenge der übermittelten Daten tatsächlich neu sind, sondern auch zu einer permanenten Rücksetzung der Formulardaten führen. Daher geht AJAX anders vor. Auch wenn die Eingabe des Textes aktiv vom Nutzer ausgeht, so veranlasst erst der Browser unsichtbar die Anfrage der Daten und ergänzt die bestehende Seite lediglich um diese. Die Kommunikation zwischen dem Browser und dem Server erfolgt demnach asynchron (**A**JAX), sieht man die Aktionen des Nutzers als eine Art „Taktgeber“ an.

Da die Daten nicht als eigenständige Webseite gesendet werden sollen, wird ein anderes Serialisierungsformat benötigt. Die Wahl fiel dabei auf XML (AJA**X**), da diese Sprache die Grundlage für XHTML darstellt und daher vielen Entwicklern bereits gut vertraut sein sollte. Auch können die meisten Browser problemlos mit XML umgehen und es existieren dafür bereits Standards zur Transformation und Verarbeitung. Allerdings kann auch jedes andere Datenformat eingesetzt werden. Da lediglich die neuen Inhalte per XML übermittelt werden und die restlichen Elemente der Seite in der Regel unangetastet bleiben sollten, muss der Browser wissen an welcher Stellt er diese einbauen muss. Hierfür wird mit DOM ein weiterer Standard bei der Webprogrammierung benützt, um den HTML Quellcode entsprechend abzuändern. [ML07]

Sollen die Anwendungen nicht nur Inhalte vermitteln, sondern auch gut bedienbar oder einfach hübsch anzusehen sein, so empfiehlt sich der Einsatz von Stylesheets. Diese Verhindern, dass die selbe Anwendung auf verschiedenen Systemen zwar des selbe Verhalten aber unterschiedliche Ausgaben erzeugt. Dabei stellt die vom W3C-Konsortium Sprache Cascading Style Sheets den wohl bekanntesten Vertreter dar. Daher ist ein Einsatz von CSS zur Formatierung der XHTML Seiten der AJAX-Anwendungen zu empfehlen. [MZ08]

Die Anwendungen können - mit Ausnahme der Zugriffe auf ein zentrales Datendepot - komplett clientseitig ausgeführt werden, indem sämtliche Funktionen in JavaScript implementiert werden. Gerade bei großen Anwendungen oder wenn die Programmlogik aus sicherheitsrelevanten Gründen nicht veröffentlicht werden soll, bietet sich dieses Vorgehen allerdings nicht an. Vielmehr sollten möglichst viele Funktionen auf dem Server ablaufen und lediglich die benötigten Ergebnisse sollten den Client erreichen. Die Wahl der hierfür eingesetzten Sprache ist dem Entwickler überlassen und sollte der Aufgabe entsprechend gewählt werden. [SF06]

CSS

(X)HTML bietet zwar stark begrenzte Möglichkeiten, um Einfluss auf die Darstellung der Inhalte zu nehmen, aber neben der Farbe, der Schriftart und dem Hintergrund sind keine weiteren Layout Veränderungen vorgesehen. Aus diesem Grund werden häufig Elemente wie Tabellen zweckentfremdet eingesetzt, um Texte auf einer Seite anzuordnen. Da diese Konstrukte für einen solchen Einsatz nicht vorgesehen sind, kann es zu unerwarteten Nebeneffekten kommen. Auch können selbst so noch lange nicht alle Layoutprobleme gelöst werden. Aus diesem Grund entwickelte das W3C mit den Cascading Style Sheets eine weitere relativ einfach zu erlernende Sprache, die es ermöglicht Inhalt (HTML) und Gestaltung (CSS) einer Seite konsequent zu trennen.

In dem HTML-Code sollte lediglich der Inhalt mit einer Gliederung hinterlegt werden. Für die Gliederung werden nur grundlegende HTML-Elemente, in der Regel ohne jegliche Attribute, benötigt. In CSS kann jedes einzelne Element oder Gruppen von gleichartigen Elementen pixelgenau angeordnet

und gestaltet werden. Des Weiteren gestattet CSS die Inhalte für unterschiedliche Zielsystemen passend für das jeweilige Gerät zu präsentieren.

Auch wenn momentan nicht jeder Browser jede Version von CSS unterstützt und eine weitere Sprache für Webanwendungen erlernt werden muss, sprechen viele Argumente dennoch für den Einsatz. So erlaubt CSS deutlich mehr Anpassungen an den Elementen vorzunehmen, als HTML-Attribute dies zulassen würden. Die strikte Trennung von Inhalt und Gestaltung ermöglicht es, Anpassungen an einem der beiden Gebieten vorzunehmen, ohne dass das andere davon beeinflusst wird. Dadurch können mehrere Layouts für Drucker, PDAs, PCs, usw. angelegt werden, die den selben Inhalt immer passend anzeigen. Schließlich verspricht die Empfehlung des W3C eine Zukunftssicherheit der Seite, da auch neuere CSS Versionen zu älteren Stylesheets abwärtskompatibel sein sollten, ohne dass größere Veränderungen an diesen vorgenommen werden müssen. [Nefo6]

DOM

Das Document Object Model stellt eine API zur Verarbeitung von gültigen HTML-Dokumenten dar. Die W3C Spezifikation von DOM wird von mehreren Sprachen, darunter auch JavaScript, unterstützt. Dabei wird die logische Struktur des Dokuments hierarchisch aufgearbeitet, um Zugriffe auf einzelne Elemente und Manipulationen an diesen zu ermöglichen. Die Vorgehensweise versteht man am besten anhand des Beispiels aus Abbildung 3.1, in dem auf der linken Seite ein Ausschnitt einer XHTML Datei dargestellt wird und auf der rechten die grafische Repräsentation des zugehörigen DOM als Baum.

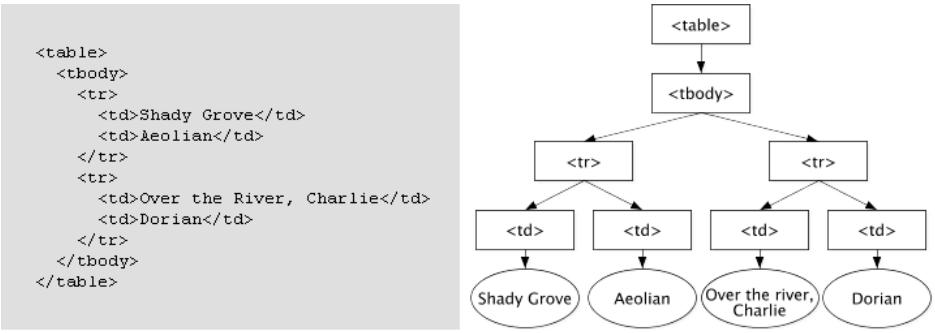

Abbildung 3.1.: Aufbau eines DOM (Quelle: vgl. [HWR04])

Da ein DOM aus maximal einem Doctype Knoten besteht unter dem ein Document Knoten angebracht wird, der seinerseits Vater von beliebig vielen Kindknoten - den Tags und deren Attribute und Werte - ist, empfiehlt sich die Darstellung als Baum. Allerdings schreibt der Standard dies keinesfalls vor. Es wird lediglich eine „geeignete“ Implementierung gefordert. In jedem Fall muss das DOM die Schnittstellen bieten, um auf die Objekte und die dazugehörigen Attribute des repräsentierten Dokuments zugreifen und bearbeiten zu können. Des Weiteren muss die logische Struktur die Beziehungen in denen die Objekte zueinander stehen aufzeigen.

Ein Zugriff auf das Dokument erfolgt über das Document Element. Von dort aus muss durch den - imaginären - Baum navigiert werden bis der zu verändernde Knoten erreicht wurde. Dieser Teilbaum kann anschließend beliebig manipuliert - also neue Knoten einfügen, bestehende Werte aktualisieren oder Knoten vollständig löschen - werden. [HWW05]

JavaScript

AJAX benötigt eine Möglichkeit das Eintreten von Ereignissen festzustellen und gegebenenfalls darauf reagieren zu können. Dies wird wie bereits erwähnt von der Skriptsprache JavaScript übernommen. Der Quellcode kann dabei direkt in die HTML Seite eingebunden werden und wird von den meisten üblichen Browsern zur Laufzeit interpretiert. Der Hauptgrund bei der Entwicklung von Webanwendungen auf JavaScript zu setzen ist der sehr einfache Umgang mit Aktionen des Nutzers. Viele Aktionen erkennt die Sprache automatisch und prüft, ob eine Reaktion darauf vom Entwickler hinterlegt wurde. Ist dies der Fall, so wird dafür gesorgt, dass der Browser entsprechend darauf reagiert. Einige der vordefinierten Aktionen, im Umfeld der Skriptsprache Ereignisse genannt, können beispielsweise auf Mausklicks, Textauswahl oder Zeigerpositionen reagieren. Dafür können Funktionen, sogenannte Event Handler, definiert werden, die einfach mit dem HTML Code interagieren können. Bei der Entwicklung größerer Funktionen sollte allerdings sehr auf eine saubere Programmierung geachtet werden, da verwendete Variablen nicht deklariert werden müssen und diese erst zur Laufzeit gebunden werden. Dadurch ist es scheinbar sehr einfach, mit der Sprache Anwendungen zu erstellen, aber meist sind sie fehleranfällig und praktisch nicht zu debuggen. Jedoch nicht nur der Umgang mit Ereignissen spricht für JavaScript als Sprache für Clientanwendungen, sondern auch die Tatsache, dass der vollständige Code beim Aufruf der Webseite übertragen wird. Dadurch ist es möglich das Programm beim Anwender ablaufen zu lassen, ohne dass es zu weiteren Netzwerkzugriffen kommt und somit der Webtraffic im Vergleich zu serverseitigen Programmen möglichst gering gehalten werden kann.

Auch wenn der Name eine starke Verwandtschaft mit Java vermittelt, so unterscheiden sich diese beiden Sprachen in sehr vielen Punkten. JavaScript ist lediglich eine Skriptsprache, die keine ausführbaren Dateien erzeugen kann oder will. Die Skripte müssen in ein Wirtssystem eingebaut werden, das sie bei Bedarf interpretieren kann. Auch wurden die objektorientierten Ansätze im Vergleich zu Java stark beschnitten. Es können weder Klassen definiert werden noch können Eigenschaften vererbt werden. Aufgrund dieser höheren Mächtigkeit von Java sollten große Anwendungen nach Möglichkeit nicht mit JavaScript, sondern mit Java umgesetzt werden. [Coh97] Dies erlauben in AJAX Frameworks wie das Google Web Toolkit (siehe Abschnitt 3.3.2).

Serverseitige Programmiersprachen

Während sich JavaScript als Standard bei der clientseitigen Programmierung durchgesetzt hat, kommen auf Serverseite mehrere unterschiedliche Sprachen zum Einsatz. Aufgabe dieser Sprachen ist es auf dem Server (X)HTML-Quellcode zu erzeugen, der daraufhin an den Client geschickt werden kann. Auch wenn Skripte, die mit diesen Sprachen entwickelt wurden, im Gegensatz zu JavaScript-Code eine dauerhafte Verbindung zum Server benötigen, um ablaufen zu können, kann dennoch bei der Entwicklung großer Webprojekte meist nicht auf sie verzichtet werden. Diese Überlegenheit resultiert beispielsweise aus der Möglichkeit mit ihnen Zugriff auf das Dateisystem des Servers zu erlangen oder Anfragen an Datenbanksysteme durchzuführen. [Wen07]

Lange Zeit bildeten diese Aufgabengebiete in Ermangelung an Alternativen die Domäne von PHP. Heute kommen allerdings auch weitere Sprachen zum Einsatz, wie Perl, JAVA, Ruby oder Python. Wichtig ist dabei in erster Linie, dass die Entwicklungszeit von neuen Funktionen in Webanwendungen mit diesen Skriptsprachen im Vergleich zu herkömmlichen, maschinennahen Sprachen erheblich verkürzt werden können. Die langsamere Ausführungsgeschwindigkeit kann nahezu vernachlässigt werden, da die Verbindungsgeschwindigkeit zwischen Server und Client ohnehin die Anwendung stärker ausbremst. [EK08]

Gerade im professionellen Umfeld kommt auf Serverseite Java in den Varianten der JavaServer Pages (JSP) und Java Enterprise Beans (JEB), zum Einsatz. Dies eignet sich immer dann, wenn es sich um sichere, stabile oder hochkomplexe Anwendungen handelt oder wenn die Webanwendungen in bestehende Infrastrukturen auf Java-Basis integriert werden sollen. Aber auch bei kleineren Projekten kann auf Java zurückgegriffen werden, da bereits sehr viele freie Erweiterungen existieren, mit denen sich Anwendungen aus den unterschiedlichsten Einsatzgebieten sehr einfach realisieren lassen. Eine optimale Zusammenarbeit mit SQL-Datenbanken ist ebenfalls gegeben.

Bei JSP wird der Code als Skript und damit Klartext in die (X)HTML Seite eingebunden und vor der Übertragung auf den Client in Hintergrund auf einem JSP-Compiler in Java-Quellcode umgewandelt. Aus diesem erzeugt ein Java-Compiler automatisch ein oder mehrere Java Servlets, die als eigenständige Java-Applikationen serverseitig ausgeführt werden können und die für den Client bestimmten dynamischen Seiten oder XML Dateien generieren. Die Möglichkeit, den Java Code direkt in den (X)HTML Code einzubinden, kann von Vorteil sein, wenn keine strikte Trennung zwischen Programmlogik und Layout erfolgen muss. Für komplexe Anwendungen empfiehlt sich allerdings eine Erzeugung der Servlets aus reinem Java-Quellcode, was mit einer konsequenten Aufspaltung von Struktur und Funktion verbunden ist. [Ste06]

Java Enterprise Beans gehen noch einen Schritt weiter, indem sie die Entwicklung verteilter, serverseitiger Enterprise Java Komponenten ermöglichen. Es werden auf spezielle Arbeitsbereiche in diesem Umfeld, wie Transaktionen, Persistenz oder Sicherheit, besonders unterstützt. Dabei unterscheiden sich JEB von vergleichbaren Technologien, in zwei Punkten entscheidenden. Zum einen versteckt die JEB Architektur die zu Grunde liegende, systemnahe Semantik nahezu vollständig, so dass der Entwickler sich nicht um die Verwaltung von Instanzen oder der Zusammenlegung von Verbindungen kümmern. Zum anderen stellt die JEB Technologie für die Geschäftslogik (Session Bean), die Persistenz (Entity Bean) und die asynchrone Kommunikation (Message Driven Bean) unterschiedliche Komponenten bereit. [Ros08] Seit der Version 1.4 stehen auch Komponenten mit Schnittstellen von Web Services (Stateless Session Beans) zur Verfügung [Wik08g].

XHTML

Die wohl bekannteste Auszeichnungssprache HTML, die zur Beschreibung von Webseiten verwendet wird, unterliegt, wie alle dieser Sprachen, der Einschränkung, dass ihre Regeln und Bestandteile nicht ohne Weiteres vom Entwickler frei gewählt werden können, sondern in einer Meta-Sprache definiert wurden. Theoretisch ist es möglich der bei HTML eingesetzten Sprache SGML neue Regeln diesem Regelwerk hinzuzufügen oder bestehende abzuändern, aber dies ist sehr kompliziert. Daher wurde mit der Entwicklung von XHTML der Versuch gestartet HTML vollständig auf der Basis von XML als Beschreibungssprache nachzubilden, da Erweiterungen in XML problemlos von der Hand gehen. Dadurch genießt XHTML alle Vorteile einer XML-Sprache, wie der Kompatibilität zu SVG, WML oder SMIL oder der Möglichkeit mittels DOM auf die einzelnen Elemente des XHTML Dokuments zugreifen zu können, kann aber dennoch als praktische anwendbare Sprache für Webseiten eingesetzt werden. Da die meisten Browser XHTML Dateien so verarbeiten, als ob sie HTML Dokumente wären, behält XHTML auch eine Kompatibilität mit alten Browsern, die ausschließlich auf HTML-Syntax ausgelegt sind. Neben der Erweiterbarkeit entfaltet XHTML ihre Mächtigkeit erst wenn beim Erstellen, Ändern oder Lesen XML-Techniken eingesetzt werden, wie es bei AJAX der Fall ist. Daher sollten neuere Browser bei der Verarbeitung und Darstellung von XHTML Dokument diese direkt als XML Dokument verarbeiten. [Red01]

XML

Der Datenverkehr im WWW benötigt ein Datenformat, das Informationen jedweder Art unter Menschen und unter Maschinen, aber auch zwischen Mensch und Maschine austauschen kann. Die E**x**tensible **M**arkup **L**anguage wurde entwickelt, um Informationen strukturiert zu speichern. Dabei wird zwischen Daten und deren Verarbeitung strikt getrennt. Die XML dient lediglich der textbasierten Datenhaltung und stellt diese jedem beliebigen Programm zur Verfügung. Dadurch wird ein sehr breites Spektrum an Anwendungsmöglichkeiten unterstützt.

Wie der Name XML bereits andeutet handelt es sich dabei um eine Auszeichnungssprache, die durch eine Kapselung in Tags den Informationen Beschreibungen über die Art der Daten mitliefert. Diese Tags sind dabei nicht, wie beispielsweise bei HTML vordefiniert, sondern können beliebig gewählt und erweitert werden. Durch eine konsequente Bindung der Daten an Tags entsteht ein hierarchische Struktur, die sich als Baum mir den Elementen als Knoten darstellen und verarbeiten lässt.

Sollen Informationen aus einer XML Datei in eine Webseite eingebaut werden, wie es beispielsweise bei AJAX erfolgt, so müssen diese zunächst in ein Format, das sich von einem Browser verarbeiten lässt,- sprich XHTML - überführt werden. Dafür stehen die E**x**tensible **S**tylesheet **L**anguage **T**ransformations, die selbst in XML formuliert werden zur Verfügung. Für die Konvertierung müssen vom Nutzer lediglich einmalig Regeln erstellt werden, wie mit den Daten der einzelnen XML Tags verfahren werden soll, um diese in das neue Format zu bringen. Ein XSLT-Prozessor führt daraufhin die Transformation automatisch durch, indem die XML Datei als Baum interpretiert wird, der nach dem gegebenen Regeln in einen Ergebnisbaum überführt wird. Bei unstrukturierten Zielformation muss dieser anschließend an das Format angepasst werden. Da in dem XML Ableger XHTML die Daten allerdings ebenfalls strukturiert als Tags vorliegen müssen, funktioniert diese Umwandlung problemlos. [WKB+03]

3.3.2. Google Web Toolkit

Grundsätzlich ist es nicht unmöglich, mit ausreichenden Kenntnissen über die Schlüsseltechnologien von AJAX, wie HTML, JavaScript, CSS und XML, eine interaktive AJAX Webseite ohne weitere Hilfsmittel zu erstellen. Jedoch ist die Zahl der Webentwickler mit einem guten Wissensstand in all diesen Sprachen sehr gering. Und selbst wenn Fachleute in den einzelnen Bereichen vorhanden sind, so ist das Management des Zusammenspiels diese Grundtechnologien sehr fehleranfällig und schwierig. Schließlich bleiben noch die Probleme bei der Verteilung der Aufgaben auf Server und Client und bei dem Zusammenspiel dieser beiden Seiten - vor allem da verschiedene Clientsysteme sich unterschiedlich verhalten können.

Das gwt[2] ist ein Ajax-Framework, das versucht, die Entwickler bestmöglich dabei zu unterstützen. Zu diesem Zweck enthält es eine große Anzahl von Tools und Bibliotheken, um die Anzahl der benötigten Sprachen in einem überschaubaren Rahmen zu halten. Die wohl wichtigste Errungenschaften besteht - neben einer einfach konfigurierbaren Schnittstelle für RPCs oder bereits vorgefertigten Widgets für die Ausgestaltung der Anwendungsoberfläche - in einem Java-zu-JavaScript-Compiler. Durch diesen ist es nicht nur möglich, serverseitige Anwendungen in Java zu programmieren - und von allen Vorteilen dieser Sprache zu profitieren - sondern auch die Skripte auf dem Client. Zusätzlich kann auch nativer JavaScript Code in den Javacode eingeflochten werden. [Ste07]

Wurden alle Funktionen vollständig implementiert und an den entsprechenden Stellen in der HTML-Seite eingebunden, so wird dem Entwickler die Möglichkeit geboten, seine Anwendung entweder

[2] kostenlos von http://code.google.com/webtoolkit/ beziehbar

im sogenannten Hosted-Mode zu testen oder für den Betrieb auf einem Server mit dem Web-Mode vorzubereiten. Der Unterschied dieser beiden Modi besteht darin, dass im Hosted-Mode der Java-Code lediglich interpretiert und in einem speziellen Browser ausgeführt wird. Dadurch arbeiten Java-Debugging Tools problemlos mit diesem Modus zusammen, was die Fehleraufspürung erheblich erleichtert. Im Gegensatz dazu setzt der Web-Mode den Java-zu-JavaScript-Compiler dazu ein, um aus dem Quellcode echten JavaScript-Code zu erzeugen. Dieser Code kann daraufhin auf einem beliebigen, Java-unterstützenden Webserver bereitgestellt werden. Dieser entstandene Code ist daraufhin auf jedem beliebigen Browser lauffähig.

Das gwt stellt aber auch Funktionen zur Verfügung, um Anpassungen für unterschiedliche Browsertypen und verschiedene Landessprachen vorzunehmen. Dabei kommt das sogenannte Deferred Binding zu tragen, das dafür sorgt, dass der vom Client zu ladende Code möglichst gering ist und nicht durch unnötige Informationen aufgebläht wird. Der Compiler erstellt daher für jede Sprache, jeden Browser, usw. eine eigene JavaScript Datei, von denen erst zur Laufzeit die benötigte bestimmt werden muss. Der so erzeugte Code ist - gerade bei großen Projekten - einem von Hand geschriebenen Code vorzuziehen. [See08]

Auf Grund der großen Beliebtheit des gwts, werden im Netz bereits eine Unzahl neuer Widgets oder Wrapper-APIs dafür angeboten. Für diese Arbeit ist besonders das Projekt GWT - Google Maps API[3] von Bedeutung. Diese Bibliothek stellt die gesamten Wrapper für das Google Map API zur Verfügung. Damit können alle Funktionen, die für die Google-Karten existieren, direkt im gwt Framework genutzt werden, ohne dass für jedes einzelne Objekt zunächst eine Wrapper-Klasse von Hand erstellt werden muss. Das Google Maps API erlaubt es in der entwickelten Webseite Weltkarten einzubauen, diese um beliebige Informationen anzureichern oder sie als Basis für weitere Funktionen der Applikation bereitzustellen. [agl07]

3.3.3. Hibernate

Das Hibernate-Projekt wird von seinen Autoren als einen mächtigen und hochperformanten Service zur persistenten Datenhaltung in relationalen Datenbanken beschrieben. Dies alleine wäre noch keine Erwähnung wert, allerdings handelt es sich bei den verwalteten Daten um Klassen, in denen Konzepte der objektorientierten Programmierung zu Einsatz kommen dürfen. Darüber hinaus bietet Hibernate eine eigene, objektorientierte Anfragesprache (HQL) als Ergänzung zu herkömmlichen SQL-Anfragen. [The09]

Alles was dafür benötigt wird sind Metadaten in denen spezifiziert wird, welche Attribute der Objekte in der Datenbank gespeichert werden sollen und wie diese in den Tabellen aufgeteilt werden sollen. Seit Hibernate 3.0 steht es dem Nutzer frei, diese Daten, wie schon bei älteren Hibernate Versionen, per externer XML-Mapping-Datei einzubinden oder dank der neu eingeführten Annotations, diese direkt an den entprechenden Stellen im Code einzutragen. Mit diesen Metadaten lassen sich aber nicht nur die Datentypen und Zeilennamen spezifizieren, sondern auch Constraints, gemäß derer Objekte validiert werden können. Auch wenn die Java Persistence API[4] vergleichbare Funktionen anbietet, ist Hibernate keinesfalls als Konkurrenzprodukt zu dieser anzusehen. Vielmehr versteht sich Hibernate als deren Implementierung, die an zahlreichen Stellen erweitert wurde. [KH06]

[3] siehe http://sourceforge.net/projects/gwt/
[4] siehe http://java.sun.com/javaee/technologies/persistence.jsp

Gilead

Während Hibernate problemlos in normalen Java-Projekten zum Einsatz kommen kann, ist dies bei der Softwareentwicklung für das gwt leider nicht möglich. Bei einem Datenaustausch zwischen Client und Server erwartet die JavaScript-Schnittstelle vollständig serialisierbare Objekte. Hibernate liefert allerdings als Ergebnismengen von Datenbankanfragen für einige Java-Typen eigene Untertypen, die das Serializable-Interface nicht implementieren. Dies tritt beispielsweise bei der Klasse der Collections auf, für die der eigene Wrapper PersistentCollection eingeführt wurde. Für einfache Sammlungen aus primitiven Datentypen lässt sich dies einfach vermeiden, indem der Server nach jeder Abfrage eine neue Sammlung erstellt und nur die Werte der PersistentCollection einfügt. Handelt es sich allerdings um Sammlungen komplexer Objekte, so ist die nur sehr schwer umsetzbar und kann schnell zu endlosen Rekursionen führen. Auch an weiteren Stellen können Probleme bei der Datenübertragung zum Client auftreten, die sich ebenfalls nur sehr schwer umgehen lassen[5].[Maro8a]

Das Gilead-Projekt wurde daher entwickelt, um diese Probleme beim Umgang mit Hibernate und Frontends, wie beispielsweise dem gwt, zu Umgehen. Dazu wurde die PersistentBeanManager-Schicht eingeführt, die jeden nicht serilisierbaren Graphen einer Hibernate-Abfrage, durch das Streichen und Ersetzen von problematischen Knoten, in einen reinen Java-Graphen überführt. Dadurch müssen lediglich sogenannte POJOs per RPC übertragen werden. [Maro8b]

3.3.4. Adobe Flash

Bei der Entwicklung der ersten Browserspiele spielte die ursprünglich von Macromedia entwickelte proprietäre integrierte Entwicklungsumgebung „Flash“, die die Erstellung von multimedialen Inhalten im Netz unterstützt, eine wichtige Rolle. Aber auch wenn diese vergleichsweise einfache Programmiersprache nicht mehr in der Lage ist, viele der neuen, komplexen Spielarten zu unterstützen, so findet sie auch heute noch bei der Programmierung von Browserspielen häufig Verwendung. Dank der sehr kurzen Entwicklungszeit und der geringen Anforderungen an das Team, können diese Titel nicht nur schnell, sondern, bedingt durch die kurze Entwicklungszeit, relativ preiswert implementiert werden. Werden die so freigesetzten Ressourcen darauf verwendet, alte Spielgenres um neue Methoden zu erweitern und diese so wieder interessanter zu gestalten, so kann Flash auch in Zukunft bei der Entwicklung von Browserspielen eine gewichtige Rolle spielen. [Auso3] Im Folgenden soll kurz auf die Geschichte von Flash eingegangen werden, bevor dieses Unterkapitel mit einem Überblick über die in Flash integrierte objektorientierte Programmiersprache ActionScript, welche Interaktivität und Navigation in Flashanwendungen erst möglich macht, abgeschlossen wird.

Flash

Der Ursprung von Flash liegt in einer reinen Zeichensoftware („SmartSketch“), die von FutureWave Software für Pen Computer - einem druckempfindlichen Display mit einem Stift als Eingabemöglichkeit - entwickelt wurde. Da sich diese Computer allerdings nicht wie erhofft auf dem Markt durchsetzen konnten, eine reine Konvertierung von SmartSketch für Windows oder Mac auf Grund der starken Konkurrenz nicht rentabel erschien, wurde nach einem neuen Verwendungszweck für den Flash-Vorgänger gesucht. Dieser wurde im aufkommenden Internet, dem bislang die Möglichkeit kleine Animationsfilme anzubieten fehlte, gefunden. Zum einen wurde eine kostenlose API in Java als Plug-in für Browser - zunächst nur für den Netscape Communicator - zum Abspielen der Filme und zum anderen ein IDE zum Erstellen von diesen entwickelt. Obwohl diese zweidimensionalen Animationen nur sehr langsam abgespielt werden konnten, kam das Produkt auf den Markt und erfuhr

[5] siehe http://hibernate4gwt.sourceforge.net/hibernate_gwt_lazy_issue.html

steigende Beliebtheit. 1996 wurde FutureWave von Macromedia aufgekauft und aus SmartSketch, das mittlerweile nicht nur einigen Namensänderungen, sondern auch Verbesserungen hinsichtlich der Geschwindigkeit erfahren hatte, wurde schließlich „Macromedia Flash“ [Waloo] ehe Macromedia - und damit auch Flash, das nun als „Adobe Flash“ vertrieben wird - 2005 von Adobe übernommen wurde [Tö5].

ActionScript

Da Flash nur als Zeichen- beziehungsweise Animationsprogramm entwickelt wurde, fehlt jegliche Möglichkeit der Interaktion, was einen Einsatz zur Spielprogrammierung unmöglich macht. Diese Lücke kann allerdings durch ActionScript geschlossen werden. Diese Programmiersprache, die mit JavaScript vergleichbar ist, arbeitet ereignisorientiert, das heißt, es wird konsequent zwischen Ereignissen (Events) und Aktionen unterschieden. Der AS-Interpreter lauscht während eine Flash-Animation abläuft, ob ein vordefiniertes Ereignis eintritt und löst daraufhin die vom Entwickler vorgegebene Aktion aus. Dabei können drei unterschiedliche Aktionsarten ausgeführt werden:

Schlüsselbild-Aktion: Befindet sich der Lesekopf des Flashplayers auf einem bestimmten Bild, so wird eine Aktion ausgeführt. Hierfür existiert es kein spezieller Event Handler.

Movieclip-Aktion: Die auszuführende Aktion ist mit einer Instanz des Films gekoppelt. Löst der Anwender während dieser ein bestimmtes Ereignis aus - beispielsweise durch einen Klick auf ein Element der Szene - so wird die Aktion gestartet.

Schaltflächen-Aktion: Die auszuführende Aktion ist mit einer Instanz einer Schaltfläche gekoppelt. Wird ein Ereignis ausgelöst - beispielsweise durch das Drücken eines Knopfes - so wird die Aktion gestartet.

Zusätzlich kann mittels eines Timers ein Event nach einem gewissen Zeitintervall ausgelöst werden. Auch wenn diese Sprache sehr trivial klingt, so können mit der Kombination aus Flash und ActionScript dennoch komplexe Webanwendungen entwickelt werden. [Volo2]

3.3.5. Zusammenfassung

Auch wenn jede der vorgestellten Webtechnologien für sich betrachtet sehr mächtig sind, so entfalten sie erst durch ein reibungsloses Zusammenspiel ihr vollständiges Potential. Die strikte Trennung von Aufbau und Darstellung der Web-Seiten durch (X)HTML mit der Ergänzungssprache CSS wurde schon lange eingesetzt und einfache Interaktionen mit dem Nutzer mittels JavaScript wurden ebenfalls schon lange realisiert [Redo1]. Auch die Verwendung von serverseitige Sprachen wie Java für den Umgang mit Dateisystemen oder relationalen Datenbanken über SQL Anfragen stellt bei der Webprogrammierung keine Neuerung dar [Weno7]. Allerdings ermöglichte erst AJAX eine flüssige Zusammenarbeit dieser Techniken, so dass ein asynchroner Datenaustausch mit dem Server als XML Daten und Manipulationen an der angezeigten Webseite, ohne dass diese vollständig neu aufgebaut werden muss, realisierbar ist [Jö8]. Das gwt stellt einen Satz von Entwicklungswerkzeugen, Programmierhilfen und Widgets, mit denen die Erstellung von Rich Internet Applikationen erheblich erleichtert werden kann [HTo7]. Setzt man AJAX in Flash-Anwendungen ein, so können diese um neue Elemente angereichert werden, wodurch dynamische und datenbankbasierte Animationen für das Web entwickelt werden können deren Einsatzgebiete praktisch unbeschränkt sind. [Muto7]

In der Tabelle 3.2 werden alle behandelten und für diese Arbeite verwendeten Sprachen und Techniken noch einmal zusammengefasst.

Sprache	Einsatzgebiet
ActionScript	Ereignisorientierte Skriptsprache für Adobe-Produkte
AJAX	Konzept der asynchronen Datenübertragung
CSS	Formatierungssprache für strukturierte Dokumente
DOM	Dokument-Modell für (X)HTML- oder XML-Dokumente
Flash	Entwicklung von animierten, interaktiven Inhalten für das Web
gwt	Framework zur Entwicklung von Webanwendungen
Hibernate	Open-Source-Persistenz-Framework für Java
Java	Serverseitige, objektorientierte Programmiersprache
JavaScript	Clientseitige Skriptsprache zur DOM-Manipulation
SQL	Anfragesprache für relationale Datenbanken
(X)HTML	Textbasierte Auszeichnungssprache
XML	Auszeichnungssprache für hierarchisch strukturierte Daten

Tabelle 3.2.: Zusammenfassung der vorgestellten Techniken der Webprogrammierung

3.4. Delivery Context: Client Interfaces (DCCI)

Aktuelle mobile Geräte stellen viele Informationen über ihren Zustand zur Verfügung - bei mobilen Geräten wären Angaben über den Batteriestand denkbar. Diese Informationen könnten zwar in den unterschiedlichsten Anwendungen sinnvoll zum Einsatz kommen - im konkreten Beispiel des Batteriestand wäre eine solche Funktion, gewisse weniger wichtige Tätigkeiten einzustellen, wenn ein bestimmter Wert unterschritten wird -, der Zugriff gestaltet sich dabei allerdings sehr schwierig, wenn von jedem Gerät eigene Schnittstellen definiert werden. Gerade im Bereich der Webprogrammierung stellt dies ein Problem dar, da die Zahl der unterschiedlichen, potentiell aufrufenden Systeme sehr groß ist und von Desktop PCs bis hin zu Mobiltelefonen reicht. DCCI wurde 2007 vom W3C eingeführt, um einen Standard in diesem Bereich zu schaffen, damit Skriptsprachen wie sie auch von dem bereits angesprochenen AJAX-Technologie genutzt werden, eine einheitliche Schnittstelle auf diese Kontextinformationen zur Verfügung gestellt wird. Im Bereich dieser Arbeit bestehen die benötigten Information aus den Positionsdaten, die von einem GPS-Empfänger ermittelt werden.

DCCI arbeitet alle erhältlichen Informationen als Document Object Model (siehe [HHW+04]) - also als hierarchischen Baum - auf. Vereinfacht ausgedrückt tauchen unter dem Wurzelelement (`Delivery Context Root`) „Properties“ mit Kontextinformationen auf, die ihrerseits als Kinder ihre einzelnen Komponenten auflisten können. In den Blätter des DCCI DOM finden sich Paare aus den Eigenschaften der Elternknoten und deren aktuell gültigen Werte. Dieser hierarchische Aufbau kann in Abbildung 3.2 nachvollzogen werden.

Dieses Modell hat neben der Tatsache, dass DOM direkt von JavaScript und damit von einer der wichtigsten Programmiersprachen im Bereich der Webprogrammierung, unterstützt wird, den Vorteil, dass im laufenden Betrieb einzelne Knoten eingefügt oder entfernt werden können.

Neben einer einfachen Navigation durch den Baum und der damit verbundenen Suche nach bestimmten Eigenschaften-Werte Paaren - im Fall dieser Arbeit die Werte des Längen- und Breitengrads, die vom GPS-Empfänger zur Verfügung gestellt werden - können, wie im DOM üblich, EventListener definiert werden, die über Veränderungen bei der Bereitstellung der Kontextinformationen informieren. Im bereits angesprochenen Beispiel des GPS-Empfängers könnte mittels Listener überwacht werden, ob ein Signal empfangen werden kann und gegebenenfalls auf eine andere Bezugsquelle

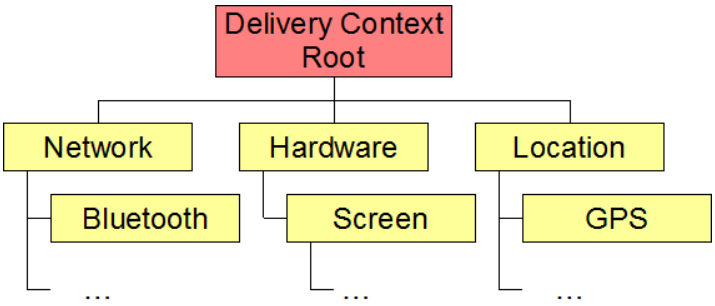

Abbildung 3.2.: Beispielhafter Aufbau eines DCCI DOMs (Quelle: vgl. [WHR+07])

der Positionsdaten umschalten. Neben den normalen in DOM definierten Ereignissen stellt DCCI einen weiteren, für die Lokalisierung sehr wichtigen, Ereignistyp bereit (`DCCI-prop-change`), der automatisch über Veränderungen in den Werten der Eigenschaften benachrichtigt. Die Spezifikation von weiteren Ereignistypen ist nicht vorgesehen und wird der jeweiligen Implementierung überlassen. [WHR+07]

3.4.1. Telar DCCI und Telar GPS

Für mobile Geräte, wie dem Nokia N810, wurde von der Open Source Community maemo.org der Browser MicroB, der auf der bekannten Mozilla Engine aufsetzt, entwickelt. Genau wie bei den restlichen Programmen der Mozilla Familie ist es somit auch bei dem MicroB Browser möglich, Erweiterungen einzubinden.

Bei Telar DCCI und Telar GPS handelt es sich um zwei solche Erweiterungen. Telar DCCI ist eine Umsetzung der in Abschnitt 3.4 beschriebenen DCCI Spezifikation. Sie stellt Anwendungen einen leeren Eigenschaftsbaum zur Verfügung, der mit gerätespezifischen Werten gefüllt werden kann. Mit dem Browser besuchte Webseiten können diesen gemäß DCCI Standard durchlaufen und Informationen daraus auslesen. Über eine Schnittstelle erlaubt Telar DCCI anderen MicroB Erweiterungen den Baum mit Inhalten zu füllen. Telar GPS ist eine solche Erweiterung, die auf die maemo Location API aufsetzt. Die durch Telar GPS und einem angeschlossenen GPS-Empfängers bestimmte GPS-Position wird anschließend an Telar DCCI weitergereicht. [Broo8]

Mit diesen zwei Erweiterungen ist es demnach möglich Webanwendungen zu entwickeln, die clientseitig auf die Daten eines GPS Empfängers zugreifen zu können und den Server somit mit den aktuellen Positionsdaten des Nutzers zu übermitteln. Dies kann über EventListener realisiert werden, die den Browser über jede Änderungen im DCCI-Baum - und somit auch bei jeder Positionsänderung - benachrichtigen. Die auf diesem Weg gewonnenen Daten können anschließend entweder direkt lokal verarbeitet werden oder für die zentrale Verarbeitung an den Server weitergereicht werden. [BNSM08]

Kapitel 4

Anforderungen an ein Framework

Nachdem nun einige existierende Pervasive Games und auch Browserspiele vorgestellt wurde, sollen im Folgenden wichtige Funktionen, die ein Großteil dieser Programme benötigen, herausgearbeitet werden. Diese Funktionen werden thematisch zusammengefasst und als Anforderungen an ein Framework formuliert. Da die Anforderungen beiden Kategorien genügen, kann dieses Framework die Entwicklung von mobilen ortsbasierten Browserspielen unterstützen, da häufig verwendete Funktionen nicht für jedes Spiel vollständig neu programmiert werden müssen, sondern maximal auf das entsprechende Umfeld angepasst werden.

4.1. Spielzustand

Ein Spiel ist grundsätzlich mit einem Zustandsautomaten vergleichbar. Innerhalb einer Spielstufe stehen jedem Spieler und computergesteuertem Objekt in jeder Spielsituation eine begrenzte, im Vorfeld definierte Anzahl an Aktionen zur Verfügung. Nach jeder dieser Aktionen verändert sich die Spielsituation. Jede Situation gleicht einem Zustand, während die Aktionen die Übergänge darstellen. Eine Spielstufe ist wiederum nichts anderes, als ein Zustandsautomat, mit einem vorgegebenen Levelanfang und je nach gewähltem Weg mindestens einem Ende. Aber auch jede der Spielstufen baut in gleicher Weise aufeinander auf. Je nachdem wie ein Level abgeschlossen wurde, wird der entsprechend nächste gestartet. Somit kann ein Spiel als Zustandsautomat, dessen Zustände wiederum aus Zustandsautomaten bestehen können, dargestellt werden.

Es wäre daher wünschenswert, dass das Framework bei definierten Zuständen und Übergängen mit vorgegebenen Start- und Endzustand, bzw. Endzuständen selbstständig und vollkommen automatisch die Spiellogik realisiert, indem intern ein Zustandsautomaten simuliert wird. Der Spielentwicklung käme ein Editor zu Gute, der es dem Nutzer erlaubt, zunächst die Spiellogik, daraufhin die einzelnen Level, und schließlich das komplette Spiel als Automat zusammenzuklicken.

[FA-1]: Das Framework muss einen Zustandsautomaten simulieren können.

4.2. Bedingungen für den Spielstart

Bereits der Spielstart kann von einem Framework unterstützt werden. Dabei soll der Spieler zunächst die Möglichkeit haben, für sich auf einem Server entweder einen neuen Account anzulegen oder auf

einen bestehenden einzuloggen. Für beide Fälle sollten entsprechende Funktionen im Framework zur Verfügung gestellt werden.

4.2.1. Benutzerregistrierung

Muss ein neuer Account erzeugt werden, sollte es die Möglichkeit geben, wahlweise eine vorgefertigte Anmeldungsseite in der die Daten, die von einem in Abschnitt 4.4 definierten Objekt für Benutzerprofile erfasst werden (Benutzername, Passwort, E-Mail-Adresse, usw.) zu verwenden oder eine eigene einzusetzen, aus deren Daten das Framework prüft, ob es einen Eintrag in einer Datenbank erzeugen kann und im Erfolgsfall dies auch ausführt. Tritt ein Fehler bei der Registrierung auf, so muss der Spieler darüber benachrichtigt werden, damit er die entsprechenden Angaben korrigieren kann.

Da es im Verlauf des Spiels nötig werden kann, mit dem Spieler in Kontakt zu treten, sollte vor dem Abschluss der Registrierung sichergestellt werden, dass mindestens die E-Mail-Adresse korrekt angegeben wurde. Aus diesem Grund muss das Framework eine Authentifizierungs-Mail erstellen können, die an die angegebene Adresse geschickt wird und einen Code zur Freischaltung des neuen Accounts enthält. Erst nach Eingabe dieses Codes wird der erstellte Account daraufhin für den Spieler freigeschaltet. Sollte die Freischaltung nicht innerhalb einer vorgegebenen Zeit erfolgen, so muss der neue Account automatisch vom Framework aus der Datenbank gelöscht werden.

[FA-2]: Das Framework muss neue Accounts anlegen können.

[FA-3]: Das Framework muss fehlerhafte Accounts löschen können.

4.2.2. Authentifizierung

Meldet sich ein Spieler mit einem bestehenden Account an, so muss eine Authentifizierung erfolgen. Erfolgte diese, so müssen die persönlichen und spielbezogenen Daten - beispielsweise die momentane Sieges- oder Niederlagenserie - des Spielers dem System zur Verfügung stehen.

Je nach Speichersystem müssen Spieler nach erfolgreicher Authentifizierung mit einem alten, gesicherten Spielstand auch die Wahl haben, entweder einen neuen Spielstand anzulegen - und damit alle alten Werte zu überschreiben - oder den alten wieder aufzunehmen. Dazu werden in Abschnitt 4.11 weitere Überlegungen angestellt. Jedoch sollte das Framework bereits automatisch beim Login die entsprechenden Funktionen aufrufen können.

Es kann aber ebenfalls vorkommen, dass ein Spieler zwar einen Account besitzt, aber sich nicht mehr an seinen Benutzername oder sein Passwort erinnern kann. Daher sollte im Framework eine Funktion existieren, die diese beiden Daten auf Wunsch an eine angegebene E-Mail-Adresse zu senden, sollte diese Adresse bei einem existierenden Account als Kontaktadresse hinterlegt sein.

[FA-4]: Das Framework muss Benutzerdaten verifizieren können.

[FA-5]: Das Framework muss einen Loginmechanismus bereitstellen.

[FA-6]: Das Framework muss dem Spieler vergessene Passworte übermitteln können.

[NFA-1]: Das Framework darf die Einbindung externer Systeme zur Benutzeridentifikation nicht verhindern.

[9] beispielsweise OpenID http://openid.net

4.3. Bedingungen für das Spielende

Für die Gruppe der Endzustände müssen weitere Funktionen zur Verfügung gestellt werden. Neben den im Vorfeld definierten Endzuständen (beispielsweise Sieg und Niederlage) kann bei dem Programm theoretisch jeder beliebige Zustand zu einem Endzustand werden, wenn die Verbindung zum Server - bewusst oder unbewusst - abbricht. Während dies bei Einzelspielerspielen zwar ärgerlich sein kann, wenn der Spielstand dadurch verloren geht, muss gerade bei kompetitiven Spielen vor der Beendigung Überlegungen hinsichtlich der Speicherung der Spieldaten getroffen werden. Je nach Genre sollte es nicht in jeder Situation möglich sein, aus dem Spiel auszusteigen, da sich ansonsten ein Spieler einen ungerechten Vorteil gegenüber seiner Mitspieler verschaffen könnte. Befinden sich beispielsweise zwei Spieler in einem Gefecht, so könnte der Unterlegene kurz vor der sicheren Niederlage das Programm beenden, ohne Konsequenzen fürchten zu müssen.

Während dies bei herkömmlichen Spielen möglich ist, indem den Spielern in diesen Situationen keine Möglichkeit gegeben wird, das Programm abzubrechen, ist dies im Browser nicht ohne Weiteres möglich, da dieser jederzeit geschlossen werden oder es zu einem unbeabsichtigten Verbindungsabbruch kommen kann. Allerdings kann mittels eines gegebenen JavaScript-Ereignisses (`onClose`) dieser Versuch festgestellt und entsprechend darauf reagiert werden. Das Framework sollte daher diesen `EventListener` bereitstellen, wobei die Implementierung der Reaktion auf den Spielabbruch dem Spielentwickler überlassen wird. Eine Speicherung der letzten verfügbaren Spieldaten im Account des Spielers sollte - abhängig vom verwendeten Speichersystem - losgelöst von der Implementierung erfolgen.

[FA-7]: Das Framework muss auf unerwartete Spielabbrüche reagieren können.

4.4. Benutzerprofil

Sollen erzielte Spielergebnisse einzelnen Spielern zugeordnet werden, so muss es nicht nur möglich sein, die Spieler eindeutig identifizieren zu können, sondern auch Daten für diese persistent zu halten. Wie man in Abschnitt 2.5 sehen konnte, erfordern unter anderem aus diesem Grund nahezu alle Browserspiele eine einmalige Registrierung des Spielers, bei der für ihn ein Benutzerprofil angelegt wird. In diesem Profil werden neben einem Benutzernamen und einem Passwort zur Identifikation des Nutzers und zur Sicherung seines Accounts auch persönliche Kontaktdaten, wie eine gültige E-Mail-Adresse, und spielbezogene Daten, wie ein persönlicher Highscore oder Speicherpunkte. In dem Framework sollte daher ein Minimalprofil mit Benutzername, Passwort, E-Mail-Adresse und Punktestand als Objekt mit den nötigen Getter- und Setter-Methoden angelegt werden, das spiel- und genreabhängig beliebig um andere Einträge erweitert werden kann.

Da momentan viele Nutzer gerade im Bereich des Internets sehr sensibel mit der Angabe persönlicher Daten sind, kann das Framework auch einen anonymen Login anbieten. Dieser würde zunächst natürlich nur ein eingeschränktes Spiel zulassen (kein Highscore, keine dauerhafte Speicherung des Spielstandes, keine Kommunikation, usw.), aber der Nutzer kann sich, ohne Sorgen um seine Daten zu haben, ein erstes Bild vom Spiel machen und daraufhin entscheiden, ob es sich lohnt einen richtigen Account zu erzeugen. Dafür muss ein Standardprofil existieren, das bereits im Framework hinterlegt werden kann.

[FA-8]: Das Framework muss ein erweiterbares Benutzerprofil zur Verfügung stellen.

[FA-9]: Das Framework muss ein anonymes Gastprofil erstellen.

4.5. Benutzerverwaltung

Existiert, wie in Abschnitt 4.4 gefordert, ein Objekt, das mehrere benötigte Daten über die Spieler enthält, so sollten weitere Funktionen im Framework bestehen, die diese Profile verwalten können. Nach der erfolgreichen Registrierung wird eine Funktion benötigt, die ein vollständiges und gültiges Profil anlegen kann. Des Weiteren kann es nötig werden, Profile zu sperren oder sogar vollständig zu löschen - sei es auf Wunsch des Nutzers oder wegen Verstößen gegen die Nutzerordnung. Auch können sich die Daten im Benutzerprofil ändern - beispielsweise der Highscore -, weshalb es dem Spiel möglich sein sollte, diese Daten direkt in das Benutzerprofil zu übernehmen. Allerdings sollte auch eine Oberfläche zur Datenmanipulation für den Spieler im Framework angelegt werden, damit dieser seine persönlichen Daten selbstständig ändern kann, wenn sich beispielsweise die E-Mail-Adresse geändert hat oder er ein neues Passwort verwenden will.

Da Browserspiele häufig unterschiedliche Spielmodi für Gratisspieler und Abonnenten anbieten, sollte das Framework diesen Status ebenfalls verwalten können. Gerade bei zeitlich begrenzten Abos, wie sie unter anderem bei Comunio aus Abschnitt 2.5.2 zum Einsatz kommen, wäre ein automatischer Statuswechsel nach Aboende oder eine Funktion zur Benachrichtigung der betroffenen Spieler, damit diese rechtzeitig über eine Verlängerung nachdenken können, sinnvoll.

Gerade Browserspiele setzen auf einen Spielprinzip, das einen Kontakt mit sehr vielen anderen Spielern benötigt (siehe Tabelle 2.4). Daher sollte es dem Spieler möglich sein, seine Kontakte zu verwalten. Dafür sollte das Framework über Funktionen verfügen, die es ihm erlauben, eine Freundesliste anzulegen, damit er bereits bei Spielstart darüber informiert werden kann, ob Spieler x ebenfalls online ist. Auch sollte es möglich sein, nicht nur private, sondern auch spielrelevante Kontakte zu pflegen, für die es möglicherweise Einschränkungen geben kann. Darunter würden beispielsweise Teams, Gilden, usw. fallen. In Abschnitt 4.10 wird näher auf die Auswirkungen dieser Zusammenschlüsse auf den Spielablauf eingegangen.

Da das Interesse an dem Spiel mit der Zeit sicherlich abnimmt, der Spieler in der Regel aber dennoch keine Notwendigkeit sieht, seinen Account wieder zu löschen, sollte die Benutzerverwaltung auch diesen Fall berücksichtigen. Das Framework kann dies relativ einfach realisieren, indem ein individueller Timer für jeden Spieler bei jedem Login dieses Spielers wieder zurückgesetzt wird. Läuft der Timer aber ab, so muss der Spieler gewarnt werden, dass dessen Account innerhalb der nächsten x Tage gelöscht wird, wenn er innerhalb dieser Zeit nicht am Spiel teilnimmt. Nach Ablauf dieser zweiten Frist können sämtliche Einträge des entsprechenden Nutzers aus der Datenbank gelöscht werden, um „Karteileichen“ zu vermeiden und das zu verwaltende Datenvolumen zu minimieren.

[FA-10]: Das Framework muss Benutzerprofile anlegen können.

[FA-11]: Das Framework muss Benutzerprofile verwalten können.

[FA-12]: Das Framework muss Benutzerprofile manipulieren können.

[FA-13]: Das Framework muss eine Oberfläche zur Profilmanipulation bereitstellen.

[FA-14]: Das Framework muss Benutzerprofile löschen können.

[FA-15]: Das Framework muss unterschiedliche Spielmodi verwalten können.

[FA-16]: Das Framework muss Benuterprofile miteinander verknüpfen können.

[FA-17]: Das Framework muss unbenutzte Benutzerprofile löschen können.

4.6. Verwaltung von geografischen Objekten

Wie bereits in Abschnitt 4.14 kurz angesprochen, können in Spielen unterschiedliche Arten von virtuellen Objekten vorkommen. Für jedes dieser Objekte muss das Framework unterscheiden können, ob es sich um ein rein statisches oder ein bewegtes Objekt handelt. Für statische Objekte wird neben dessen Position auch die Möglichkeit gegeben sein, diese mit einer Eigenschaft bei einem Kontakt mit dem Spieler zu versehen. Mögliche Werte für diese Eigenschaft wären sammelbar - verbunden mit dadurch ausgelösten Funktionen - oder undurchdringlich, oder sie dienen als Auslöser für weitere Ereignisse. Neben rein statischen Objekten kommen in Spielen in der Regel aber auch bewegte Objekte zum Einsatz. Für diese muss es möglich sein zu unterscheiden, ob das Objekt einem vorgegebene Pfad folgt oder eine eigene KI besitzt. Für ersteres müssen Funktionen angeboten werden, die die einzelnen Wegpunkte einlesen, die Geschwindigkeit der Objekte ermitteln und Auslöser für die Bewegung festlegen können. Bei KI-Objekten muss das Framework eine Schnittstelle anbieten, die dem Spielentwickler erlaubt, die Logik, der diese Objekte folgen, anzugeben. Allerdings wird in der Realisierung des Frameworks von der Implementierung nicht-statischer Objekte abgesehen.

[FA-18]: Das Framework muss statische Objekte verwalten können.

[FA-19]: Das Framework muss Objekte darstellen können.

[FA-20]: Das Framework muss Objekten Eigenschaften zuweisen können.

4.7. Ressourcenverwaltung

Da kein Spiel ohne Ressourcen auskommt, sollte sich ein Spiel-Framework auch mit der Verwaltung dieser kümmern. Neben den Ressourcen „Punkte“ und „Zeit“ mit denen sich die Abschnitte 4.8 und 4.9 beschäftigen, können genreabhängig sehr viele weitere, wie Gold, Holz, usw., hinzukommen. Die benötigten Funktionen zur Verwaltung dieser Rohstoffe können in drei Gruppen eingeteilt werden:

4.7.1. Allgemeine Ressourcenverwaltung

Die Funktionen der allgemeinen Ressourcenverwaltung betreffen sowohl jene, die sich im Besitz des Spielers befinden, als auch die, die dem Spiel „gehören“. Zunächst muss ein Objekt „Ressource“ angelegt werden können, das bereits vom Framework mit einigen Grundeigenschaften, wie beispielsweise ihrem Namen und ihrer Position, ausgestattet wird und vom Entwickler um weitere Eigenschaften - wie beispielsweise dem Gewicht - erweitert werden. Das Objekt muss ebenfalls eine Methode beinhalten, dass es „verbraucht“ werden kann, das bedeutet in erster Linie, dass das Objekt gelöscht wird. Da aber eventuell durch den Einsatz weitere Objekte erzeugt werden oder andere Funktionen dadurch aufgerufen werden sollen, muss auch diese Methode erweiterbar sein.

Häufig unterstützt ein Spiel auch einen An- und Verkauf von Rohstoffen. Dazu muss geklärt werden, welchen Wert die einzelne Ressource hat. Dies ist in der Regel aber kein statischer Wert, sondern er orientiert sich beispielsweise an Angebot und Nachfrage im Spiel selbst, wie es bei Comunio aus Abschnitt 2.5.2 der Fall ist, oder an realen Umrechenkursen. Daher sollte eine Schnittstelle existieren, über die eine Funktion eingebunden werden kann, die diesen Wert bestimmt. Des Weiteren sollte ein Objekt „Bank“ im Framework existieren, der die Spieler überschüssige Ressourcen verkaufen oder von der fehlende erworben werden können.

[FA-21]: Das Framework muss erweiterbare Ressourcen verwalten können.

[FA-22]: Das Framework muss alle Ressourcentypen ermitteln können.

[FA-23]: Das Framework muss eine allgemeine abstrakte „Währung“ einführen.

[FA-24]: Das Framework muss eine Schnittstelle zur Implementierung von „Wechselkursen“ besitzen.

[FA-25]: Das Framework muss die Ressourcen aus den Benutzerprofilen auslesen können.

[FA-26]: Das Framework muss Ressourcen zwischen der virtuellen Welt und den Spielern austauschen können.

4.7.2. Verwaltung von Ressourcen der virtuellen Welt

Die Ressourcen, die sich im Besitz des Computers befinden, müssen in regelmäßigen Abständen erzeugt werden. Dies kann entweder automatisch nach einer vorgegebenen Zeit geschehen oder von einem Ereignis ausgelöst werden. Daher sollte dies als Interface im Framework hinterlegt werden, bei dem vom Spielentwickler die Bedingung zur Generierung implementiert werden müssen.

Damit der Spieler diese generierten Rohstoffe erwerben kann, müssen ebenfalls Regeln angegeben werden können, unter welchen Bedingungen dies vonstatten gehen darf. Auch dies muss als Interface realisiert werden, da es dafür je nach Spiel viele unterschiedliche Anforderungen geben kann. Im Framework können aber Grundanforderungen implementiert werden, beispielsweise dass sich der Spieler an dem selben Ort wie die Ressource befinden muss. Da nach Verbrauch der Ressourcen diese wieder in den Besitz des Computers übergehen, muss ebenfalls geregelt werden können, ob diese endgültig „verbraucht“ wurden oder von anderen Spielern wieder erworben werden können.

[FA-27]: Das Framework muss Ressourcen erschaffen und verbrauchen können.

4.7.3. Verwaltung spielereigener Ressourcen

An die Verwaltung der Ressourcen, die sich im Besitz eines Spielers befinden, werden weitere Anforderungen gestellt. So muss geklärt werden, ob es für eine Ressource eine Beschränkung gibt, wie beispielsweise eine maximale Traglast. Dies ist aber nicht zwangsläufig ein konstanter Wert, sondern kann sich von Spieler zu Spieler unterscheiden. So ist es möglich, dass ein Anfänger im Spiel nur sehr wenig Gewicht tragen kann, während erfahrenere Spieler deutlich mehr aufnehmen dürfen. Ein anderes Beispiel einer solchen Beschränkung wäre, dass eine Ressource nur in Kombination mit einem anderen Gegenstand erworben werden kann. So könnte es sein, dass Holz nur dann gewonnen werden kann, wenn der Spieler im Besitz einer Axt ist. Daher muss auch eine Schnittstelle existieren, um solche Anforderungen einbringen zu können. Diese Anforderungen berühren auch jene an die computereigenen Ressourcen, aber da das Hauptaugenmerk dieser Beschränkungen auf den bereits erworbenen Gegenständen liegt, werden sie der Verwaltung spielereigener Ressourcen zugerechnet.

Für bestimmte Aktionen wird häufig der Besitz eines gewisses Rohstoffs vorausgesetzt. Daher muss das Framework ebenfalls Anfragen hinsichtlich der vollständigen Ressourcen jedes Spielers bearbeiten können. Auch wenn diese Anforderung sehr trivial klingt, kann diese Verwaltungsaufgabe sehr komplex sein, wenn nicht nur individuelle Ressourcen, sondern auch Team-Ressourcen, wie in Abschnitt 4.10.2 beschrieben, beachtet werden müssen.

[FA-28]: Das Framework muss Ressourcen unter den Spielern austauschen können.

4.8. Punktestand

Da es in jedem Spiel gilt, einen Punktestand zu verwalten, sollte das Framework auch hierfür Funktionen bereitstellen. Bei den Punkten kann es sich allerdings neben „echten“ Punkten auch um andere Ressourcen, wie Zeit, Rohstoffe usw. handeln. Daher wird, um unterschiedliche Leistungen vergleichbar zu machen, zunächst eine Umrechenfunktion in eine vergleichbare Einheit, beispielsweise in den Warenwert der Rohstoffe, benötigt. Auf Grund der erzielten Spielergebnisse sollte ebenfalls eine Highscoretabelle angelegt werden können, in der der endgültige Punktestand jedes Spielers hinterlegt wird, da auf diese Weise aus nahezu jedem klassischen Einzelspieler ohne weiteres ein pseudo Mehrspielerspiel gemacht werden kann und durch den Ansporn der beste Spieler zu werden weiterer Spielspaß generiert wird. Auch sollte der Punktestand exportiert werden können, um diesen auf einem der in Abschnitt 4.13 vorgestellten Kommunikationswegen an andere Spieler zu übermitteln.

[FA-29]: Das Framework muss den Punktestand jedes Spielers in der Spielwährung ermitteln können.

[FA-30]: Das Framework muss eine Highscoretabelle erstellen können.

[FA-31]: Das Framework muss den Punktestand exportieren können.

4.9. Zeit

Die Zeit spielt bei den Spielen an unterschiedlichen Stellen eine entscheidende Rolle. So kann sie als Punktestand eingesetzt werden, wie es bei Yetisports (siehe Abschnitt 2.5.6) in einigen Disziplinen der Fall ist. Je schneller eine Aufgabe gelöst oder je länger eine Aufgabe fehlerfrei ausgeführt wird, desto mehr Punkte müssen dem Spieler gutgeschrieben werden. Dafür muss ein Timer zu Aktionsbeginn gestartet werden können, der zu einem definierten Ende stoppt.

Eine andere Möglichkeit, Zeit in ein Spiel einfließen zu lassen, findet sich in Uncle Roy All Around You (siehe Abschnitt 2.4.12). Dem Spieler steht die Zeit als eine Art Ressource zur Verfügung, die, wenn sie vollständig aufgebraucht ist, eine Aktion, wie beispielsweise das Spielende auslöst. Im Gegensatz zu einem normalen Timer, der zu Spielbeginn einfach abläuft, sollte es für diesen Zweck auch möglich sein, dem Zeitkonto wieder Zeit als Belohnung gutzuschreiben.

Während die oben genannten Einsatzgebiete sich ausschließlich auf die Spielzeit beziehen, kann auch die Echtzeit gerade für die sogenannten Endlosspiele, die im Onlinesektor sehr beliebt sind, von Interesse sein. So können beispielsweise Ressourcen nach einer vorgegebenen Zeit wieder „nachwachsen“, wie es in Travian (siehe Abschnitt 2.5.4) der Fall ist. Diese Zeit muss unabhängig davon, ob Spieler auf dem Server eingeloggt sind, immer weiterlaufen und zu vorgegebenen Zeitpunkten Ereignisse auslösen.

Schließlich gibt es noch eine weitere Funktion der Zeit. In der Spieleserie Boktai, die in Abschnitt 2.3.5 kurz eingeführt wurde, können bestimmte Aktionen nur zu bestimmten Tageszeiten ausgeführt werden. Während Boktai die „Zeit“ lediglich über die Helligkeit ermittelt, kann dies auch über die Echtzeit erfolgen. Diese Funktion sollte allerdings nur in Einzelspielerspielen eingesetzt werden, da ansonsten eine gemeinsame Zeit für alle Zeitzonen bestimmt werden müsste, die nichts mehr mit der echten Tageszeit der Nutzer zu tun hat. Würde die Zeit der jeweiligen Zeitzone herangezogen werden, so könnte es zu ungerechten Duellen kommen, da den Spielern unterschiedliche Aktionen zur Verfügung stehen.

Spielunabhängig müssen im Bereich der Spielzeiten aber auch gesetzliche Bestimmungen beachtet werden. So gelten beispielsweise in China bereits seit dem Jahr 2005 Richtlinien, die die maximale

Spielzeit von Computerspielen jeder Art auf maximal drei Stunden beschränken, um die davon ausgehende Suchtgefahr eindämmen zu können [Golo5]. Ein Framework sollte auch aus diesem Grund die individuelle Spielzeit der Spieler beobachten und ein Interface bereitstellen, das den Entwicklern erlaubt, nach einer vorgegebenen Zeit entsprechende Schritte - beispielsweise eine Mitteilung an den Spieler oder der sofortige Logout - einzuleiten.

Daher sollte das Framework die Spielzeit, sowie die Echtzeit verwalten können, um timergesteuerte Ereignisse auslösen zu können. Neben einer Funktion, die einen Timer erstellt, sollte das Framework ein Zeitkonto verwalten, echtzeitgesteuerte Ereignisse definieren und unterschiedliche Zeitabschnitte festlegen können.

[FA-32]: Das Framework muss die aktive Spielzeit der Spieler bestimmen können.

[FA-33]: Das Framework muss die gesamte Spielzeit der Spieler bestimmen können.

[FA-34]: Das Framework muss das „Alter“ der virtuellen Objekte bestimmen können.

[FA-35]: Das Framework muss die aktuelle Spielzeit bestimmen können.

[FA-36]: Das Framework muss mit timergesteuerten Ereignissen umgehen können.

4.10. Teamverwaltung

Da in Mehrspielerspielen nicht nur gegeneinander, sonder häufig auch miteinander gespielt wird, muss ein gutes Framework auch für diese Fälle Funktionen bereitstellen. Neben losen Gruppen, die nur dem Zweck dienen, den individuellen Erfolg sicherzustellen, kommt es in vielen Spielen auch zu festen, dauerhaften Bindungen unter den beteiligten Spielern. Der Sinn dieser Gruppierungen besteht nicht nur darin, die normalen Einzelspieler-Spielziele zu erreichen, sondern auch im Austausch von Ressourcen oder es werden weitere Aufgaben für die Mitglieder freigeschaltet, wie dem Kampf gegen andere Gruppen. Für diese Art von festen Gruppen, auch oft als Team oder Gilde bezeichnet, werden daher in zwei Bereichen weitere Anforderungen an das Framework gestellt. So muss es sich um die Administration der einzelnen Mitglieder kümmern und die gemeinsamen Team-Ressourcen verwalten können.

[FA-37]: Das Framework muss Teams als Spielerobjekte verwalten können.

[FA-38]: Das Framework muss Benutzerprofile von Teamspielern erweitern können.

4.10.1. Mitgliederverwaltung

Auch wenn in Abschnitt 4.5 bereits die Verwaltung der einzelnen Spieler geregelt wurde, so müssen für den Eintritt in ein Team weitere Punkte beachtet werden. Zunächst muss für jede Gruppe ein Teamstatus erstellt werden, in dem Informationen - wie die Mitglieder, deren Aufgabe im Team, die Team-Ressourcen oder den Rang des Teams - hinterlegt werden kann.

Auch für den Eintritt in eine Gruppe müssen bestimmte Anforderungen erfüllt werden. Im Gegensatz zum Eintritt in ein Spiel, soll sich nicht jeder Spieler bei jeder beliebigen Gruppe anmelden können. Vor dem Beitritt muss eine Funktion überprüfen, ob dieser Spieler bereits Mitglied einer andern Gruppe ist. Ist dies nicht der Fall, so kann es spielabhängig zu weiteren Bedingungen an den Spieler kommen. Daher soll dem Entwickler ein Interface bereitgestellt werden, damit dieser solche Bedingungen erstellen kann. Eine wichtige Art der Anmeldung sollte aber bereits implementiert werden. Bei dieser ist die Anmeldung erst dann möglich, wenn man entweder einer leeren Gruppe

beitritt, diese neu erstellt, oder wenn man von Mitgliedern einer existierenden Gruppe eingeladen wird.

In der Regel wird einem der Mitglieder die Aufgabe das Leiters übertragen. Dafür muss ein Wahlsystem existieren, damit die Spieler abstimmen können, wer diese Aufgabe übernehmen darf. Als Leiter besitzt der Spieler nicht nur die Möglichkeit, Teamentscheidungen zu treffen, wie dem Eintritt in eine Gruppenaufgabe, sondern er besitzt auch administrative Aufgaben innerhalb der Gruppe. Diese beinhalten beispielsweise auch den Ausschluss von Teammitgliedern. Somit hat diese Wahl auch Auswirkungen auf die individuellen Status aller Mitglieder.

[FA-39]: Das Framework muss eine Mitgliederverwaltung anbieten.

4.10.2. Team-Ressourcenverwaltung

Auch hinsichtlich der Ressourcen müssen für Teams zusätzlich zu den in Abschnitt 4.7 geforderten Funktionen weitere bereitgestellt werden. Neben der reinen Verwaltung von erreichten Prämien, beispielsweise durch den erfolgreichen Abschluss von Teamaufgaben, sollte es den einzelnen Mitgliedern auch möglich sein, dem Team einen Teil der eigenen Ressourcen zur Verfügung zu stellen und im Bedarfsfall aus der gemeinsamen Kasse wieder zu entnehmen. Während die Einzahlung ohne Bedingung erfolgen kann, sollte es für die Entnahme möglich sein, gewisse Anforderungen zu stellen, wie beispielsweise die Zustimmung durch den Teamleiter, damit sich kein Mitglied hemmungslos bedienen kann. Von einer automatischen Zusammenlegung aller individueller Ressourcen sollte abgesehen werden, da ansonsten ein Austritt eines Mitglieds aus einem Team große Probleme an die Ressourcenverteilung stellen würde. Des Weiteren muss das Framework bei der Team-Ressourcenverwaltung den Entwicklern gestatten, die Team-Ressourcen auf eine Teilmenge aller im Spiel vorkommender Ressourcen zu beschränken oder diese um spezielle Team-Ressourcen zu erweitern.

4.11. Zwischenspeicherung

Wie man bei den Browserspielen sehen konnte (siehe Tabelle 2.5), kommen grundsätzlich drei verschiedene Techniken zum Einsatz, um eine Spielunterbrechung zu ermöglichen und dennoch den Spielfortschritt nicht zu verlieren. Im Wesentlichen werden für die Realisierung aller Unterarten der drei Ansätze Funktionen für

- ein ereignisgesteuertes, automatisches Speichersystem,
- ein allgemeines, manuelles Speichersystem,
- ein kontextabhängiges, manuelles Speichersystem und
- kein Speichersystem

im Framework benötigt. Dem jeweiligen Speichersystem muss lediglich bei der Implementierung angegeben werden, welche Spieldaten zu sichern sind. Die Abbildung der oben genannten Ansätzen auf die Speichersysteme ist in Tabelle 4.1 dargestellt.

[FA-40]: Das Framework muss den aktuellen Spielzustand persistent halten können.

[FA-41]: Das Framework muss alte Spielzustände wiederherstellen können.

[FA-42]: Das Framework muss kontextabhängig die Speicherung erlauben oder verbieten können.

Speicherart	Speichersystem
manuelle, vollständige Speicherung	allgemein
Logout-Speicherung	ereignisgesteuert
Savepoints	kontextabhängig
Passwortsystem	kein Speichersystem
keine Speicherung	kein Speichersystem

Tabelle 4.1.: Abbildung der unterschiedlichen Speicherarten auf die Speichersysteme

4.11.1. Vollständige Speicherung

Zunächst sollte eine vollständige Speicherung ermöglicht werden. Dabei müssen alle relevanten Daten (Position, eingesammelte Gegenstände, Anzahl der „Leben“, Punktestand, usw.) in einer Datenbank, dem Spieler zugeordnet, hinterlegt werden, damit die Spielwelt nach Wiederaufnahme des Spiels unverändert ist. Dabei lassen sich zwei Systeme unterscheiden. Zum einen kann die Speicherung vom Spieler manuell angestoßen werden. Dies erweist sich gerade bei kompetitiven Mehrspielerspielen als unpraktisch, da ansonsten im trial-and-error Verfahren der optimale Lösungsweg erforscht werden kann. Zum anderen kann das Spiel beim Ausloggen automatisch gespeichert und nur beim Einloggen geladen werden. Dadurch entfällt der Vorteil des fehlerfreien Spiels, da, wurde der optimale Lösungsweg einmal verlassen, nicht auf diesen zurückgekehrt werden kann. Ein weiteres Problem bei dieser Art der Speicherung tritt im Zusammenhang mit mobilen ortsbasierten Spielen auf. Werden die Positionsdaten nicht relativ, sondern absolut verwendet, kann die Speicherung, beziehungsweise das Laden selbiger nicht erfolgreich durchgeführt werden. Daher sollte entweder von der Kombination aus diesem Speicherverfahren und der Verwendung der absoluten Positionsdaten Abstand genommen werden oder in diesem Fall eine Speicherung der Ortsdaten verhindern.

4.11.2. Savepoint Speichersystem

Dieses Problem kann - zumindest teilweise - umgangen werden, indem man nur an bestimmten Orten in die Spielwelt eintreten, beziehungsweise diese nur dort verlassen kann. Dafür muss ein sogenanntes Savepoint Speichersystem angeboten werden. Dabei spielen die Positionsdaten nur eine Rolle, um zu überprüfen, ob hier eine Spielunterbrechung erlaubt ist, aber sie müssen nicht zwangsläufig erhalten werden. In vielen Spielen wird dieses System noch weiter vereinfacht, indem nicht nur an bestimmten Orten, sondern auch erst nach dem Erfüllen bestimmter Aufgaben gespeichert werden darf. Somit wäre es möglich, endlich viele Eintrittspunkte direkt im Spiel zu definieren, an denen bereits feststeht, welche Gegenstände der Spieler hat, welche Missionen abgeschlossen wurden, usw. Handelt es sich hierbei um kein kompetitives Spiel, so ist ein Passwortsystem denkbar, das nach jeder erfolgreich ausgeführten Aufgabe dem Spieler einen Code übermittelt, der es ihm erlaubt, jederzeit an dieser Stelle sein Spiel fortzusetzen. Auf jeden Fall sollte zumindest das Framework diese Möglichkeit anbieten.

4.11.3. Kein Speichersystem

Eine letzte Möglichkeit mit einer Spielunterbrechung umzugehen ist, diese vollständig zu verbieten, beziehungsweise mit einer sofortigen Beendigung des Spiels gleichzusetzen. Ist die Spielzeit entsprechend kurz und der Spielcharakter nicht auf den Wettkampf mit anderen Spielern ausgelegt, dann

ergibt dieses System durchaus Sinn und hat seine Berechtigung. Daher muss ein entsprechender Ansatz im Framework realisiert werden.

4.12. Persistenz

Wie man in Kapitel 2.5 gesehen hat, und wie bereits in Abschnitt 4.2.1 beschrieben wurde, erfordert nahezu jedes Browserspiel eine Registrierung der Spieler, bei der mehrere spielerbezogene Daten persistent gespeichert werden müssen. Im Framework sollten daher ebenfalls Funktionen implementiert werden, die die häufigsten Anfragen für all diese Daten unterstützen.

Neben den bereits angesprochenen Bereichen der Benutzeranmeldung beziehungsweise des Logins und der Speicherung der Spielstände sollte daher auch Anfragen hinsichtlich des momentan aktuellen Highscores bereitgestellt werden, wie es in Abschnitt 4.8 gefordert wurde. Auf diese Weise wird der Spieler motiviert, mehr Leistung zu erbringen, um den derzeitigen Spitzenreiter zu überflügeln. Findet diese Anpassung nicht zeitnah zu Verbesserungen anderer Spieler an dem Höchststand statt, stellt sich unter Umständen schnell eine Enttäuschung bei den Nutzern ein, da sich eine sicher geglaubte Führung auf Grund von veralteten Werten schnell als falsch herausstellen kann.

Schließlich müssen in diesem Zusammenhang auch Vergleiche mit den Mitspielern möglich sein. Dazu muss sichergestellt werden, dass auf bestimmte Spieldaten jederzeit zugegriffen werden kann, auf andere je nach Beziehung zu dem anderen Nutzer - beispielsweise sollten Spieler innerhalb der selben Gruppe jederzeit über die Ressourcen aller Mitglieder informiert sein, Gegner dürfen diese Werte aber nicht kennen - und auf die privaten Daten auf keinen Fall. Das Framework sollte daher vor der Ausführung einer Anfrage prüfen, welche Einträge aus dem Benutzerprofil für den jeweiligen Anwender sichtbar sein dürfen.

Nicht nur die Persistenz der Daten muss bei einem mobilen ortsbasierten Browserspiel sichergestellt sein, sondern gerade bei Endlosspielen muss auch die Spielwelt persistent sein, unabhängig davon, ob Spieler in diese eingeloggt sind oder nicht. Daher sollte das Framework Funktionen anbieten, die die Welt dauerhaft am Leben halten. Das heißt, dass sich virtuelle Objekte auch nach dem Logout weiterbewegen und agieren können oder Ressourcen dauerhaft produziert werden.

[FA-43]: Das Framework muss Benutzerdaten persistent halten können.

[FA-44]: Das Framework muss Spieldaten persistent halten können.

[FA-45]: Das Framework muss Daten geheim halten können.

4.13. Kommunikationsmöglichkeiten

Da nahezu alle Browserspiele durch den Aufbau einer Community versuchen, Spieler möglichst dauerhaft an das Spiel zu binden, sollten im Framework Möglichkeiten dafür bereitgestellt werden. Ein wichtiger Identifikationspunkt wäre ein Forum, in dem die Spieler einander sowohl beim eigentlichen Spiel, als auch bei sonstigen Themen helfen können und Tipps geben. Auf diese Weise werden nicht nur soziale Bindungen aufgebaut, sondern es stärkt auch das Zusammengehörigkeitsgefühl zu der Seite, beziehungsweise dem Spiel, da man sich für seine Mitspieler verantwortlich fühlt. Ein solches Forum kann bereits vorgefertigt in das Framework eingebaut werden und ohne große Anpassungen in jedes Spiel übernommen werden.

Benötigt ein Spieler in einer Spielsituation akut Hilfe, so erweist sich der Weg über das Forum häufig als zu langsam. Da anzunehmen ist, dass auch bei Einzelspielerspielen mehrere Spieler gleichzeitig

auf dem Server eingeloggt sind, würde sich daher ein einfacher Chat Room anbieten. Die dafür benötigten Komponenten (Texteingabeleiste und Ausgabefenster) sollten daher ebenfalls für den Einsatz in den Spielen im Framework hinterlegt werden.

Während die ersten beiden Möglichkeiten unabhängig von der Spielart sind, werden bei Mehrspielerspielen weitere Kommunikationsmittel benötigt. So sollte im Framework neben dem allgemeinen Chat auch ein Chat mit einem eingeschränkten Bereich, den nur die Mitspieler oder sogar nur eine Teilmenge dieser betreten und lesen können. Dabei sollte neben speziellen Chats für die unterschiedlichen identifizierbaren Gruppen (siehe Teamverwaltung 4.10) auch ein Chatmodus für alle Spieler in der „Nähe“ existieren. Dieser letztgenannte Modus soll ein normales Gespräch simulieren, wenn das Spiel relative Positionsdaten verwendet. In diesem Fall können sich zwei Avatare begegnen, obwohl die Spieler weit voneinander entfernt sind.

Schließlich sollte auch ein Mailsystem implementiert werde, damit sich die Mitspieler auch auf diesem Weg kontaktieren können. Dafür können entweder die wirklichen E-Mail-Adressen der Benutzer, die bei der Benutzerregistrierung angegeben wurden, ausgetauscht werden oder pseudo Adressen generiert werden, die nur innerhalb des Spiels gültig sind. Durch letzteres kann die Anonymität der Nutzer gewahrt werden, ohne dass im Spiel auf ein Mailfunktionen verzichtet werden muss.

[FA-46]: Das Framework muss ein Forum anbieten.

[FA-47]: Das Framework muss einen Chat anbieten.

[FA-48]: Das Framework muss Mailfunktionalitäten besitzen.

4.14. Spielfeldeditor

Ein Spielfeldeditor muss zwei wesentliche Funktionsweisen unterstützen. Bei mobilen ortsbasierten Spielen, denen ein rein virtuelles Modell zu Grunde liegt, muss der Spielfeldeditor Funktionen anbieten, die vergleichbar mit einem Zeichenprogramm sind, um die Karten oder Level gestalten zu können. Zusätzlich müssen sich diese Karten auf ein Koordinatensystem abbilden lassen, um eine Steuerung über die Positionsänderung zu unterstützen.

Neben der reinen Oberfläche muss der Editor auch Funktionen anbieten, die spielrelevante Objekte identifizieren können, um diese mit einer Funktion zu versehen. Beispiele hierfür wären Wände, sammelbare Gegenstände oder Gegner. Dabei müssen zwei unterschiedliche Typen unterschieden werden: spielrelevante Punkte und spielrelevante Gebiete. Für Punkte muss eine Funktion bereitgestellt werden, die GPS-Daten entgegennimmt und in das Modell übernimmt. Im konkreten Fall würde dies bedeuten, dass Daten entweder eingegeben oder über die DCCI ausgelesen und daraufhin an das kartographische System weitergereicht werden. Bei Gebieten werden zwei entgegengesetzte Eckpunkte benötigt, um einen rechteckigen Bereich zu bestimmen.

Auch wenn diese Methode für umfangreiche Karten ausgesprochen aufwendig ist, führt für die virtuellen Objekte an dieser Arbeit kein Weg vorbei. Sollen allerdings reale Gebiete und Gebäude im Spiel erkannt werden, für die es bereits in externen Quellen Positionsdaten gibt, so sollten diese eingebunden werden können. Im Framework sollte daher ein Interface angeboten werden, das diese Daten ausliest. Die konkrete Implementierung, inklusive Zugriff auf die Quelle und Umwandlung der Daten, muss aber für jede Quelle vom Programmierer von Hand durchgeführt werden.

Für mobile ortsbasierte Spiele, die auf einem Modell der realen Welt stattfinden, existiert die Spielumgebung bereits oder kann über ein Kartenmodul zugeladen werden. Zusätzlich müssen aber dennoch die oben beschriebenen Funktionen zur Erfassung spielrelevanter Orte und Objekte verwendet werden, um die existierende Karte um virtuelle Aspekte anreichern zu können.

[FA-49]: Das Framework muss einen Editor anbieten.

4.15. Positionsdatenermittlung

Bei mobilen ortsbasierten Browserspielen spielen die Positionsdaten in jedem Fall eine entscheidende Rolle. Daher muss das Framework in diesem Umfeld zwei wichtige Aufgaben übernehmen. Zunächst muss eine Funktion existieren, die die Positionsdaten des GPS-Signals über DCCI auslesen kann und daraufhin zur Verfügung stellt. Für diese Einträge muss ein `EventListener` definiert werden, damit sie immer auf dem aktuellsten Stand gehalten werden können.

Die Spielprogrammierung kann aber mit einer zusätzlichen Funktion noch weiter unterstützt werden. Diese Funktion sollte den Abstand des Spielers, beziehungsweise seines Avatars, zu spielrelevanten virtuellen Objekten bestimmen können. Zu „virtuellen Objekten“ zählen in diesem Fall sowohl rein virtuelle Objekte, als auch virtuelle Modelle realer Orte. Diese Funktion wird benötigt, um schnell zu ermitteln, ob ein Gegenstand „eingesammelt“ wurde, ob ein bestimmter Bereich betreten wurde, ob man sich in der Nähe eines Mitspielers befindet, usw. Da dies für Spiele aller Arten benötigt wird, sollte eine solche Funktion daher auch im Framework hinterlegt werden.

[FA-50]: Das Framework muss die Verfügbarkeit eines GPS-Signals ermitteln können.

[FA-51]: Das Framework muss auf Positionsupdates reagieren können.

[FA-52]: Das Framework muss die Position des Spielers ermitteln können.

[FA-53]: Das Framework muss die Position von Objekten der Spielwelt beziehen können.

[FA-54]: Das Framework muss den Abstand des Spielers zu den Objekten der Spielwelt ermitteln können.

[FA-55]: Das Framework muss weitere ortsabhängige Ereignisse definieren.

4.16. Koordinatensysteme

Da es alleine für die geographische Länge und Breite mehrere Darstellungsmöglichkeiten gibt, und neben diesem auch weitere Koordinatensysteme existieren, sollte ein Interface erstellt werden, in dem Umrechnungsformeln hinterlegt werden können, wenn für das Spiel eine andere Darstellungsform benötigt wird, als die von dem GPS gelieferte. Dies kann immer dann der Fall sein, wenn die absolute Position des Spielers in das Spiel übernommen werden soll, also wenn sich der Spieler in der realen Welt an der Position (x | y) befindet, so steht auch seine Spielfigur in der virtuellen Welt genau an diesen Koordinaten. Bei einem relativen Einsatz der Daten - das heißt, bewegt sich der Spieler x Meter nach Norden, so bewegt sich auch sein Avatar um x Einheiten nach Norden - müssen die Daten aber ebenfalls aufgearbeitet werden. Dazu müssen in der Funktion des Frameworks aufeinander folgende Positionsdaten hinterlegt und lediglich die Änderung an das eigentliche Spiel weitergeleitet werden, bzw. diese an die Koordinaten der virtuellen Welt angepasst werden.

[FA-56]: Das Framework muss Koordinaten unterschiedlicher Systeme umrechenen können.

4.17. Interaktion mit verschiedenen Multimediaformaten

Für die Spiele kann es auch von Interesse sein, Multimediaformate einzubinden. Dabei kann es sich beispielsweise um Audiodateien, Filme oder Flashanwendungen handeln. Damit diese auch auf die Spielsituation eingehen, beziehungsweise diese beeinflussen können, muss ein Datenaustausch zwischen dem Spiel und dieser Anwendung stattfinden können. Ein mit dem Framework erstelltes Projekt muss je Format eine aus- und eine eingehende Methode anbieten, um dies zu ermöglichen. Da es unmöglich ist, für alle webfähigen Datenformate eine bereits vorgefertigte Schnittstelle im Framework zu implementieren, muss dies vollständig dem Spielentwickler überlassen werden. Eine Blockade eines dieser Formate seitens des Frameworks sollte allerdings ausgeschlossen werden.

[NFA-2]: Das Framework darf das Einbinden von Multimediaformaten nicht verhindern.

4.18. Minispiele

In sehr vielen Browserspielen (beispielsweise Travianer aus Abschnitt 2.5.5) kommen kleine Minispiele zum Einsatz. Diese Spiele können entweder Einfluss auf das eigentliche Spiel haben oder zusätzlichen Spielspaß generieren. Thematisch und inhaltlich müssen sie nicht mit dem Hauptspiel in Zusammenhang stehen, weshalb sie unabhängig von diesem Entwickelt werden können.

In einem mobilen ortsbasierten Browserspiel wäre ein Minispiel von Vorteil, wenn beispielsweise kein GPS-Signal verfügbar ist oder kein WLAN-Hotspot erreicht werden kann. Das Framework könnte (sehr einfache) Minispiele anbieten, die direkt in das eigentliche Spiel übernommen werden können. Neben den Spielen muss eine Funktion zur Verfügung gestellt werden, die erlaubt, diese direkt aufzurufen.

4.18.1. Fragenkatalog

In einigen Spielen wird ein Quiz als Minispiel eingesetzt. Auch wenn dies nicht sehr häufig vorkommt, sollte dennoch ein Fragenkatalog, in dem neben den Fragen auch die Antwortmöglichkeiten und die Reaktionen auf die unterschiedlichen Eingaben verwaltet werden. Ein solcher Katalog kann, wenn er sehr allgemein gehalten wird, nicht nur als Minispiel, sondern auch für alle anderen Anfragen an den Nutzer - beispielsweise „Brauchen Sie eine Anleitung?“ - eingesetzt werden.

Im Framework soll eine Oberfläche hinterlegt werden, die aus einem Datenbanksystem die Fragen und Antworten einliest, und die Auswahl des Nutzers zurückliefert. Darauf wird die in der Datenbank angegebene Funktion als Reaktion auf diese Antwort ausgeführt. Diese Funktionen sollten nicht im Framework implementiert werden, um die Mächtigkeit des Fragenkatalogs nicht einzuengen.

Ebenfalls sollen Funktionen zur Verfügung gestellt werden, um diese Fragen anlegen zu können. Dafür müssen Datenbankeinträge angelegt und verwaltet werden können. Diese Funktionen werden bereits bei der Erstellung des Spiels benötigt, sie können aber auch Elemente des eigentlichen Spiels sein, wenn beispielsweise der Gewinner einer Quizrunde die nächsten Fragen stellen darf.

[FA-57]: Das Framework muss ein erweiterbares Flash Quiz einbinden können.

[NFA-3]: Das Framework darf die Kommunikation zwischen den Minispielen und dem Hauptspiel nicht verhindern.

Kapitel 5

Das Framework

Nachdem nun hinreichend geklärt wurde, was von dem zu entwickelnden Framework erwartet wird und welche Funktionalitäten es zu erfüllen hat, beschäftigt sich dieses Kapitel mit dessen Aufbau. Dazu wird zunächst in Abschnitt 5.1 auf das gewählte Schichtenmodell zur Kapselung der Client-, Server- und Datenbankaufgaben eingegangen. In Abschnitt 5.2 wird daraufhin die Paketstruktur erläutert, die diesem Framework zu Grunde liegt, bevor sich Abschnitt 5.3 mit deren Inhalte beschäftigt.

Auch wenn die Wahl eines geeigneten Datenbankschemas für die Performanz des Frameworks äußerst entscheidend ist, so wird an dieser Stelle dennoch nicht weiter darauf eingegangen, da die Tabellen vollständig von Hibernate (siehe Abschnitt 3.3.3) erstellt wurden. Für die Abbildung der Objekte des Frameworks auf die Tabellen der Datenbank wird daher auf Anhang B verwiesen.

5.1. Die Schichtenarchitektur

Wie aus den in Kapitel 4 erläuterten Anforderungen an ein Framework für mobile ortsbasierte Browserspiele rasch ersichtlich ist, werden sowohl clientseitige Komponenten - beispielsweise die Oberflächen für das Login - als auch serverseitige - wie die Verarbeitung der Spiellogik in dem Zustandsautomaten - benötigt. Des Weiteren müssen Datenobjekte persistent gehalten werden können, was einen Einsatz eines Datenbanksystems nahezu unumgänglich macht.

Die Wahl eines angemessenen Architekturschemas fiel daher auf ein klassisches Drei-Schichten-Modell mit einer clientseitigen Präsentationsschicht, einer serverseitigen Schicht zur Realisierung der Spiellogik und einer Datenhaltungsschicht, deren Aufgaben von einem beliebigen DBMS übernommen werden. Durch dieses Architekturschema sind beispielsweise Migrationen auf andere Datenhaltungssysteme oder die Einbindung in eigene Widgets ohne größere Umstände möglich. Auch werden so die sensiblen Nutzerdaten besser geschützt, da clientseitige Zugriffe auf diese nur über Anfragen an den Server erfolgen können und kein direkter Kontakt mit der Datenbank besteht. Eine strikte Trennung der Funktionen, die die Anwendungsdaten verarbeiten, von denen, die sie zur Verfügung stellen, kann des Weiteren zu einer besseren Lastverteilung auf dem Server führen. [Ramoo]

Abbildung 5.1 stellt das gewählte Architekturschema dar. Neben den drei bereits erläuterten Schichten wird hier allerdings die Anwendungsschicht nochmals in drei seperate Bereiche unterteilt. Sämtliche RPCs eines gwt Widgets werden von der gwt-Service-Schicht entgegengenommen und beantwortet. Die eigentliche Verarbeitetung der Anfragen findet allerdings erst in der Spiellogikschicht statt.

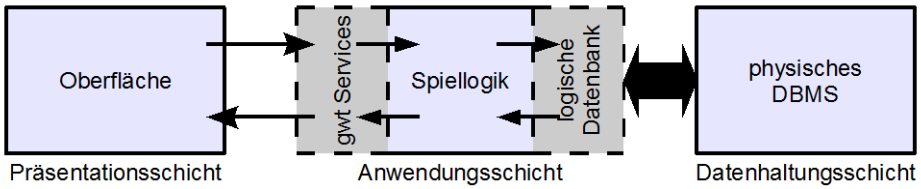

Abbildung 5.1.: Drei-Schichten-Modell des Frameworks

Müssen Objekte persistent gelagert oder abgerufen werden, so stellt die logische Datendank für jedes Objekt sogenannte Datenzugriffsobjekte (DAO) - in Anlehnung an die gleichnamige Microsoftschnittstelle - zur Verfügung. Mit diesen sind Abfragen an ein Datenbanksystem möglich. Die Auswertung der Daten erfolgt allerdings ausschließlich in der Spiellogikschicht. Dadurch müssen für die Verwendung von anderen Oberflächen oder Datenbanksysteme lediglich diese beiden Schnittstellen verändert werden, die eigentliche Logik bleibt davon aber unberührt.

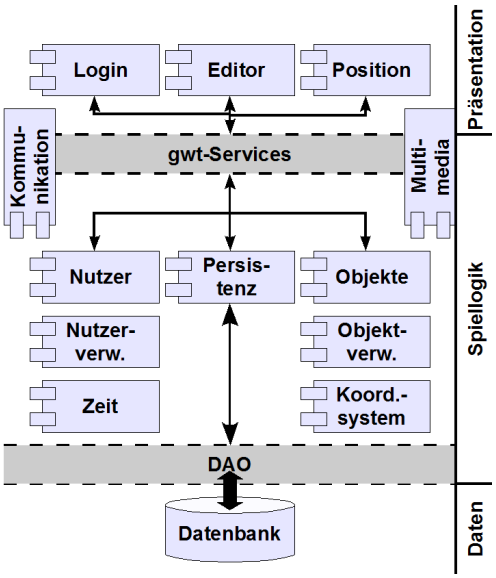

Abbildung 5.2.: Schichtenmodell des Frameworks

Nachdem die Architektur nun bestimmt wurde, müssen die geforderten Softwarekomponenten einem der drei Bereiche zugeteilt werden. Abbildung 5.2 stellt diese Zuordnung in einer vereinfachten Form dar. Dabei werden vergleichbare Elemente - beispielsweise Nutzerdaten, Spielerdaten, Teamdaten - zur Erhöhung der Übersichtlichkeit miteinander verschmolzen.

Der Datenschicht wird keine der Komponenten zugeordnet, da diese nur für die persitente und konsistente Haltung der Daten zuständig ist und automatisch von einem DBMS übernommen werden kann. Auch müssen, wie bereits in der Einleitung dieses Kapitels beschrieben, keine Tabellen von Hand angelegt werden, da diese automatisch von Hibernate erzeugt werden können.

Der Großteil aller benötigten Komponenten liegt in der Anwendungsschicht. Zu diesen Komponenten zählen neben jeglichen Spielobjekten - beispielsweise Spieler, Gegenstände, besondere Orte, usw. - und

deren Verwaltung auch jene Komponenten, die die Verwendung und Manipulation von Koordinaten- - beispielsweise Abstandsberechnungen oder Umrechnungen in unterschiedliche Koordinatensysteme - und Zeitdaten - - ermöglichen. Das Herzstück dieser Schicht stellt allerdings die Persistenzkomponente dar, die die Ergebnisse der restlichen Komponenten zusammenträgt und verarbeiten kann. Auf die genauen Kommunikationsmöglichkeiten der Komponenten innerhalb der Schicht soll an dieser Stelle jedoch nicht eingegangen werden - und wurde daher nicht in dem Modell aufgenommen -, da es dafür eine genauere Betrachtung der einzelnen Klassen benötigt. Jedoch erfolgt die Kommunikation mit der Schnittstelle zur Datenhaltungsschicht stets über die Persistenzkomponente. Diese fordert Daten aus der Datenbank an, bereitet die erhaltenen Werte auf und stellt sie den restlichen Komponeneten der Anwendungsschicht daraufhin zur Verfügung. Auch werden die Anfragen, die die Clients mittels gwt Services stellen von der Persistenzschicht beantwortet. Allerdings benötigen die Widgets auch direkten Zugriff auf die entwickelten Spielobjektklassen, damit sie von diesen Instanzen erzeugen können. Daher ist auch ein Kontakt Client-Nutzer - und damit auch Client-Spieler, Client-Team - oder Client-Objekt - inklusive aller abgeleiteter Klassen - zulässig.

Auf der Clientseite werden sämtliche Oberflächen, wie der Loginmanager inklusive aller von ihm benötigter Unterwidgets - beispielsweie für die Spielerauswahl - oder der Editor ausgeführt. Die Aufgabe dieser Oberflächen besteht allerdings nur darin, Eingabedaten zu sammeln und in Datenobjekten zu verpacken, die über definierte Serviceschnittstellen zur eigentlichen Verarbeitung an die Persistenzkomponente weitergeleitet werden können. Auch die Ermittlung der aktuellen Position muss vom Client ausgeführt werden können, da nur dieser Zugriff auf die Daten der entsprechenden Hardwarekomponente hat.

Während sich alle bislang beschriebenen Komponenten relativ eindeutig den einzelnen Schichten zuordnen lassen, so existieren mit der Kommunikations- und Multimediakomponenten zwei Einheiten, die sowohl einen serverseitigen Anteil zur Verarbeitung der Logik, als auch einen clientseitigen Anteilt zur Darstellung besitzen. Die Untrennbarkeit dieser beiden Funktionsweisen rührt zum einen daher, dass für die Kommunikationmittel auf vorgefertigte Lösungen verwiesen werden muss[1], da eine Eigenentwicklung auf Grund des zeitlichen Rahmens dieser Arbeit nicht möglich war. Zum anderen ist bei den Multimediaanwendungen der Ort der Ausführung von der zum Einsatz kommenden Sprache abhängig, so dass keine Festlegung auf client- oder serverseitige Ausführung erfolgen sollte, um die offene Architektur des Framework nicht unnötig einzuschränken.

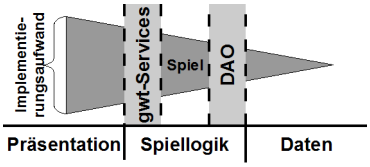

Abbildung 5.3.: Implementierungsaufwand für ein mobiles ortsbasierts Browserspiel

Da das Thema dieser Arbeit mobile ortsbasierte Browserspiele beinhaltet, stellt sich unweigerlich die Frage, wo diese in dem Architekturschema angesiedelt werden können und welche Anpassungen möglicherweise dafür nötig sind. Abbildung 5.3 stellt dies in Abhängigkeit zu dem zu erwartenden Implementierungsaufwand dar. Während das Framework nur wenige Oberflächen anbietet, da diese sich zu sehr von Spiel zu Spiel unterscheiden, und daher viel Eigenentwicklung von Nöten ist, sollte die Datenschicht nahezu unverändert übernommen werden können. Allerdings können Veränderungen in der Datenschicht, wie beispielsweise eine Erweiterung der Attribute eines Spielerobjekts, Anpassungen an der entsprechenden Datenbank benötigen. Dieser Eingriff kann zwar über

[1] z.B. http://www.board-4you.de/ für ein Forum oder http://www.web-tech-india.com/software/jsp_chat.php als Beispiel einer Chat Applikation

die Mapping-Konfiuration erfolgen, jedoch ist dieser Aufwand dennoch der Datenhaltungsschicht zuzurechnen.

5.2. Die Pakete des Frameworks

Im Folgenden sollen nun die zum Einsatz kommenden Module des Frameworks vorgestellt werden. Die statischen Struktur wird in dem Paketdiagramm aus Abbildung 5.4, das sich an der UML-Notation orientiert, wiedergegeben.

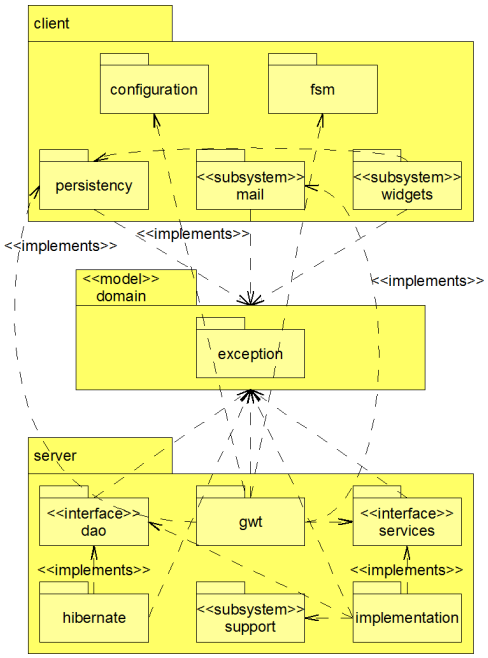

Abbildung 5.4.: Paketdiagramm des Frameworks

Neben dem `client`-Paket, in dem üblicherweise alle clientseitigen Daten von gwt hinterlegt sind, kommen bei dem Framework zwei weitere große Pakete zum Einsatz: das `domain`-Paket und das `server`-Paket. Unter dem `server`-Paket finden sich alle Klassen, die zur Verarbeitung der Spiellogik benötigt werden. Sie realisieren dabei einen Großteil der Logikschicht aus Abschnitt 5.1 inklusive der Schnittstellen zu einer Datenbank und den Clients. Das `domain`-Paket stellt die von beiden Seiten gemeinsam genutzten Objekte zur Verfügung. Der genauere Aufbau dieser drei Oberpakete soll im Folgenden näher beleuchtet werden.

5.2.1. Das `client`-Paket

Innerhalb des `client`-Pakets wurden die Klassen des Frameworks in fünf Paketen zusammengefasst, von denen zwei auf Grund ihrer Abschlossenheit und Unabhängigkeit vom restlichen System als eigene Untersysteme angesehen werden können. Im Einzelnen finden sich hier folgende Pakete:

`configuration`: Hier befinden sich alle zur Initialisierung und Konfigurierung des Servers benötigten Funktionen. Diese müssen einmalig vom Client beim Start der Anwendung ausgeführt werden, um RPCs zu ermöglichen. Auch wenn es auf Grund dieses stark beschränkten Einsatzgebietes denkbar wäre, diesen Funktionen kein eigenes Paket zu geben, sondern den Code an der benötigten Stelle einzufügen, so wurde im Framework dennoch die dargestellte Struktur verwendet, um eine bessere Übersicht und einfachere Portierbarkeit zu erzielen.

`fsm`: Für den Zugriff auf die Zustände des simulierten Automatens wurde alle benötigten Methoden in diesem Paket hinterlegt. So ist es den Clients beispielsweise möglich, die augenblicklich zulässigen oder alle bekannten Kommandos, abzurufen oder auszuführen.

`mail`: Im `mail`-Paket wurden alle Klassen, die für den Versand von Nachrichten benötigt werden, zusammengefasst. Da diese Funktionen nahezu unabhängig vom restlichen System sind - und daher unverändert auch in anderer Software, wie beispielsweise WebMail Oberflächen zum Einsatz kommen könnten - werden sie als eigenes in sich geschlossenes Untersystem angesehen. Lediglich auf die Objektdeklarationen - und damit auf das `domain`-Paket - benötigen diese Komponenten einen Zugriff. Dieser ließe sich aber durch eine leichte Anpassung - beispielsweise durch die Übermittlung der Nutzer-Id anstelle des kompletten Nutzer-Objekts - umgehen.

`persistency`: In dem `persistency`-Paket findet man alle Funktionen, die für den Austausch von Spielobjekten zwischen Client und Server zu deren Speicherung, Wiederherstellung, Manipulation oder endgültigen Löschung, benötigt werden. Dazu müssen die Klassen dieser Objekte bekannt sein, weshalb ein Import des `domain`-Pakets unumgänglich ist.

`widgets`: Auch wenn das Hauptaugenmerk dieses Frameworks auf der Bereitstellung von einfach wiederverwendbaren Funktionen und Schnittstellen der Spiellogik liegt, so wurden dennoch einige allgemein einsetzbare Widgets für gwt erschaffen. Diese sind dabei so weit losgelöst vom restlichen Framework, dass auch dieses Paket als eigenes Untersystem betrachtet werden kann. Jedoch werden auch von dieser Einheit die Objektdeklarationen benötigt, um beispielsweise eine Maske zur einfachen Erstellung neuer Nutzer anbieten zu können. Da diese Daten auch dauerhaft in der Datenbank übernommen werden sollten, muss das `widget`-Paket neben dem Zugriff auf `domain` auch Zugriff auf die Schnittstellen des `persistency`-Pakets erhalten.

Wie man bereits an dieser Stelle sehen kann, handelt es sich bei den clientseitigen Objekten und deren Methoden in erster Linie um Schnittstellen für den Datenaustausch mit einem Server. Weitere clientseitige Pakete werden in dem Framework aber auch nicht benötigt, da diese in der Regel zu spezifisch auf ein Spiel angepasst werden müssen, weshalb dies dem Spieleentwickler überlassen werden muss.

5.2.2. Das `domain`-Paket

In dem `domain`-Paket wurden alle Klassen und Pakete zusammengefasst, die keine eigene Logik besitzen und lediglich Deklarationen für das übrige System bereitstellen. Aus diesem Grund werden auch keine Referenzen auf andere Komponenten des Frameworks benötigt. Nichtsdestotrotz stellt dieses Paket dennoch einen großen Nutzen für das komplette System dar, was durch die hohe Anzahl an Verweisen auf dieses Paket leicht sichtbar ist.

Neben den Klassen, die direkt in dem `domain`-Paket hinterlegt sind - dabei handelt es sich um die Deklarationen aller Objekte, die im Framework definiert werden, wie Nutzer oder Spieler, aber auch die Zustände des Automaten - befindet sich auch das Unterpaket `exception` in diesem Paket. Darin befindet sich eine Sammlung von Ausnahmen, die speziell auf die Bedürfnisse des Frameworks angepasst sind. Dadurch ist bespielsweise eine genauere Fehleranalyse - verbunden mit für den

Nutzer verständlichen Ausgabewerten - möglich. Beispielsweise kann ein Fehler bei der Eingabe einer ungültigen Emailadresse geworfen und mit einer entsprechender Nachricht darauf reagiert werden.

5.2.3. Das `server`-Paket

Wie bereits im Schichtenmodell in Abbildung 5.2 ersichtlich wurde, werden die meisten der Komponeneten des Frameworks auf dem Server ausgeführt. Daher ist auch das `server`-Paket das größte der drei Hauptpakete. Die hier implementierte Logik teilt sich in sechs Unterpakete auf:

`dao` und `hibernate`: Das `dao`-Paket stellt Interfaces für den Zugriff auf die persistenten Daten in einer Datenbank zur Verfügung - aus diesem Grund müssen die Klassen jener Objekte aus dem `domain`-Paket importiert werden. Es werden hier keine spezifischen Anfragen erlaubt, sondern nur die, die als unbedingt nötig angesehen werden. Das bedeutet, dass beispielsweise alle Objekte eines bestimmten Typs ermittelt werden können, aber eine Prüfung auf ein bestimmtes Attribut an dieser Stelle nicht möglich ist. Dadurch werden teilweise unnötig viele Datensätze aus der Datenbank geladen. Diese Einschränkung kann gerade bei einer sehr großen Datenbank zu langen Ladezeiten führen. Dennoch ist dieser Schritt nötig, da das Paket die DAO-Schicht realisieren soll. Diese Schicht kennt die Implementierung der eigentlichen Systemlogik nicht. Somit muss ein Kompromiss zwischen implementierter, aber nicht benötigter und nicht angebotener, aber gewünschter Methoden getroffen werden.

Da die DAO-Schicht nicht nur system-, sondern auch datenbankunabhängig arbeiten soll, befinden sich in diesem Paket ausschließlich Interfaces. Dadurch kann problemlos auf eine andere Datenbank migriert werden, ohne dass an den Komponenten des Servers - deren Anfragen sich ausschließlich an Objekte des `dao`-Pakets richten - oder am `dao`-Paket selbst Anpassungen vorgenommen werden müssen. Lediglich eine Implementierung der Schnittstellen pro Datenbanksystem wird benötigt.

In diesem Framework kommt Hibernate zum Einsatz (siehe Abschnitt 5.4). Daher werden in dem `hibernate`-Paket beispielhafte Implementierungen mit HQL-Anfragen bereitgestellt.

Benötigt werden diese Pakete nur von dem `services`-Paket - bzw. dessen Implementierung. Dadurch kann sichergestellt werden, dass die Clients nur die Daten erhalten, die für sie bestimmt sind, da auf dem Server eine Filterung vor der Weiterleitung erfolgen kann.

`gwt`: Während von dem `dao`-Paket die DAO-Schnittstelle realisiert wird, setzt das `gwt`-Paket die gwt-Services-Schicht um. Daher befinden sich im `gwt`-Paket die Implementierungen der clientseitigen RPC-Interfaces aus den Paketen `configure`, `mail` und `pesistency`. Da sich diese Aufrufe auch auf Objete des Systems beziehen, muss das `gwt`-Paket ebenfalls Klassen aus der `domain` importieren.

Auch wenn im Schichtenmodell die gwt-Services-Schicht nur als Kommunikationskomponente zwischen Client und Server gesehen wird, gehen die von diesem Pakets übernommenen Aufgaben ein wenig hinaus. So übernimmt es einfache Transformationen und Prüfungen der eingehenden Daten, um unberechtigte Service-Anfragen bereits frühzeitig zurückweisen zu können. Zulässige Anfragen werden an die serverseitigen Services weitergeleitet, die sich in dem Paket `services` befinden.

`services` und `implementation`: In dem `service`-Paket liegen alle Funktionen, die die Daten zwischen dem `gwt`-Paket, also den Anfragen des Clients, und dem `dao`-Paket, dem Kontakt zu der Datenbank, austauschen. In diesem Paket wird die Spiellogik umgesetz. Daher findet hier nicht nur eine Verknüpfung zwischen den beiden Schnittstellen der Logikschicht statt, sondern alle

Daten werden, entsprechend der Implementierung, angepasst, angereichert und manipuliert. Des Weiteren müssen hier auch alle eingehende Daten aus der Datenbank gefiltert werden können, da auf diese, wie bereits bei der Beschreibung des `dao`-Pakets erklärt, nur mittels Standardanfragen zugegriffen werden kann.

Um das Framework an dieser Stelle nicht unnötig einzuschränken, werden die Services zunächst nur als Schnittstellen angeboten. Daher müssen die `service`-Interfaces lediglich das `domain`-Paket importieren, während das `dao`-Paket nicht benötigt wird, da auch eine Bearbeitung der Anfragen ohne Datenbank denkbar ist. Eine beispielhafte Implementierung der Schnittstellen, bei der die Hibernate-Datenbank verwendet wird, befindet sich in dem `implementation`-Paket. Dieses Paket muss daher neben dem `domain`-Paket auch die `dao`-Schnittstellen importieren. Des Weiteren wird auch auf Funktionen des `support`-Pakets zurückgegriffen auf deren Bewandtnis im folgenden Punkt näher eingegangen wird.

`support`: Im letzten Paket wurden alle bei der Umsetzung der Spiellogik unterstützenden Funktionen, die selbst aber nicht mit dieser zusammenhängen, zusammengefasst. Dazu zählt beispielsweise ein XML-Parser, mit dessen Hilfe unter anderem die Implementierung der Mailkomponente konfiguriert werden kann, oder eine Bibliothek mit Umrechnungsvorschriften für Koordinatensysteme. Da diese Funktionen zum einen auf Grund der offenen Implementierung keinen direkte Bindung zu dem restlichen Framework besitzt und zum anderen jede darin befindliche Klasse in sich abgeschlossen ist, wird dieses Paket im Modell als Subsystem gekennzeichnet.

Wie man leicht erkennen kann, wurde die Schichtenarchitektur konsequent bei der Wahl der Pakete beachtet und umgesetzt. Auch wurde durch den Einsatz von Interfaces an den Schnittstellen der Schichten eine leichte Portierbarkeit auf andere Datenbanksysteme oder eine andere Benutzeroberfläche ermöglicht, was den ohnehin offenen und erweiterbaren Aufbau des Frameworks nicht unnötig einschränkt.

5.3. Die Klassen des Frameworks

Nachdem alle abstrakten Einheiten des Frameworks vorgestellt und beschrieben wurden, kann nun auf deren Inhalte - den Klassen und Interfaces des Frameworks - eingegangen werden, um dessen Funktionsweise und Mächtigkeit darlegen zu können. Dadurch sollen denkbare Einsatzmöglichkeiten vermittelt werden. Zu diesem Zweck kommen für die wichtigsten Klassen Klassendiagramme[2], die an die UML-Spezifikation angelehnt sind, zum Einsatz. Dabei können die Klassen des `client`-Pakets vernachlässigt werden, da diese nur die Widgets zur Realisierung einer vollständigen Login-Oberfläche, einer Editor-GUI[3], Eingabemasken für die Datenmanipulation und einer Komponente für die Einbindung von Flash-Anwendungen bereitstellen oder Schnittstellen für den Datenaustausch mit einem Server definieren, deren Implementierung in dem `gwt`-Paket in Abschnitt 5.3.3 besprochen wird.

5.3.1. Die Klassen des `domain`-Pakets

Wie bereits in Abschnitt 5.2.2 erläutert, ist der Inhalt des `domain`-Pakets zweigeteilt. Es besteht zum einen aus den Klassendefinitionen der Spielelemente und zum anderen aus der Implementierung eines endlichen Automaten. Ebenfalls werden hier die neu eingeführten `exceptions` definiert.

[2] in gekürzter Form ohne getter, setter oder für die interne Realisierung benötigte Methoden
[3] aus Zeitgründen wurde diese Komponente nicht implementiert

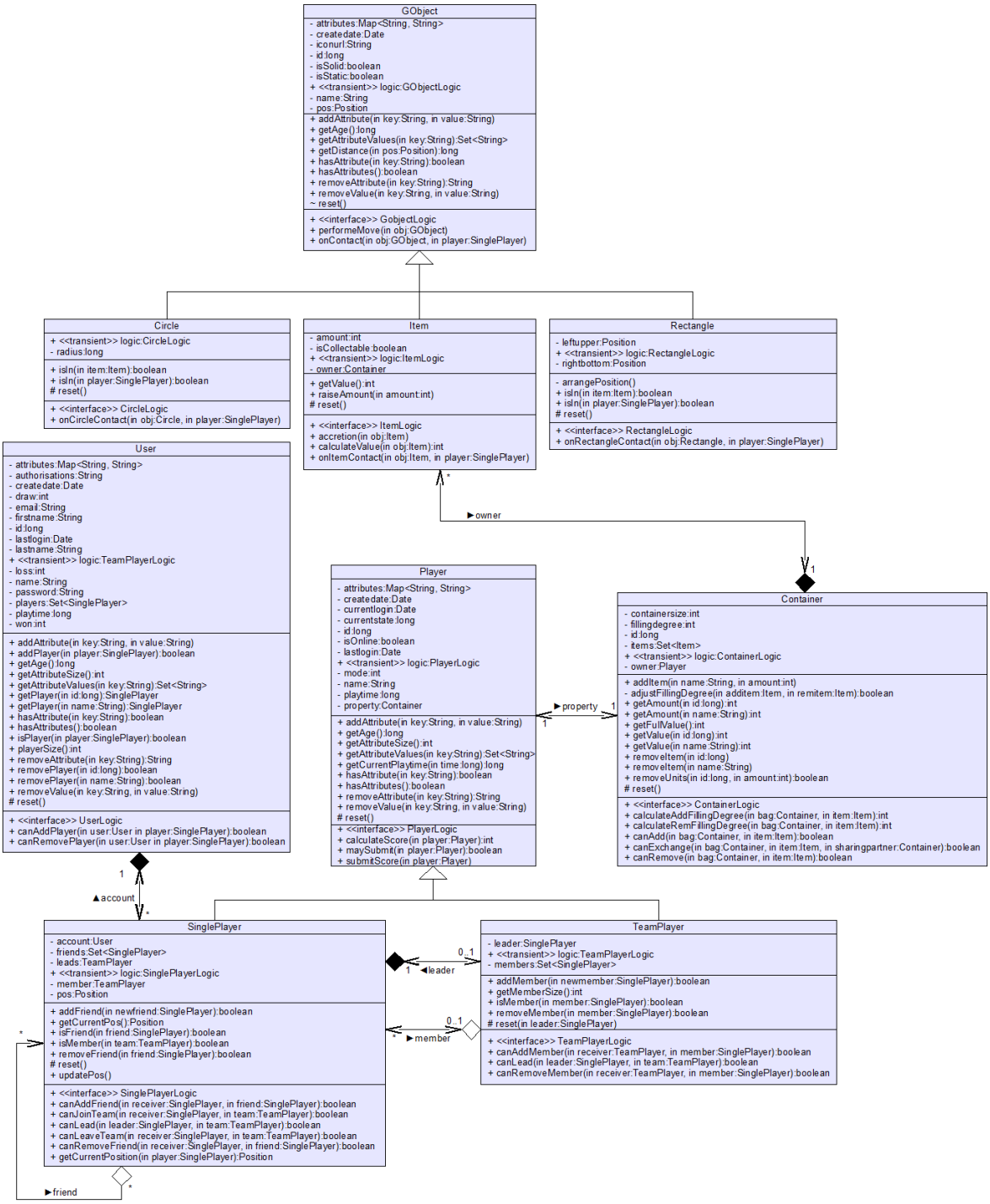

Abbildung 5.5.: Die Klassen des `domain`-Pakets - die Spielklassen

Abbildung 5.5 bildet die Hierarchie und die Assoziation der Klassen aller Spielobjekte ab. Da ein Großteil der vorkommenden Attribute, Referenzen und Methoden selbsterklärend und deren Nutzen

für das Framework leicht ersichtlich ist, wird im Folgenden lediglich auf die wichtigsten Aspekte eingegangen. Nahezu jede der dargestellten Klassen besitzt oder erbt das Attribut `attributes`, durch das eine Erweiterbarkeit gegeben ist. Hier können weiter benötigte Objekteigenschaften definiert und mit beliebig vielen Werten belegt werden, die, in einem einzigen String kodiert, auf den Eigenschaftsnamen gemappt werden. Auch das Datum an dem das jeweilige Objekt angelegt wurde, findet sich in diesen Klassen, um das virtuelle Alter der Objekte bestimmen zu können.

Bei einer Registrierung wird für jeden neuen Nuter ein Objekt der `User`-Klasse angelegt und die entsprechenden Attribute wie Vorname, Nachname, usw. mit den Nutzerdaten belegt. `authorisations` definiert die Rolle des Nutzers. Vom Framework sind die Rollennamen `ADMINISTRATOR` für unbeschränkte Rechte, `GUEST` für starkbeschränkte Rechte und `USER` für den normalen Betrieb vorgesehen. Da eine mögliche Einschränkung in einer vorgegebenen, maximalen Spielzeit bestehen könnte, wird für jeden Account die Spielzeit (`playtime`) als Summe der Spielzeiten jedes erstellten, zum `User` gehörenden, Spielers gespeichert. Ein Datumsfeld, in dem der Zeitpunkt des letzten erfolgreichen Logins gespeichert wird (`lastlogin`), erleichtert die Wartung der Datenbank, da so „tote“ Accounts automatisch gelöscht werden können. Da ein `User`-Objekt lediglich der Speicherung der Nutzerdaten dient, benötigt jeder Account ein `SinglePlayer`-Objekt, um einem Spiel beitreten zu können. Diese Objekte werden in dem `players` Attribut als Set gespeichert, damit das Framework nicht unnötig restriktiv ist und nur ein Spielerobjekt pro Account erlauben würde. Die offene Architektur des Frameworks gestattet es allerdings, über die `UserLogic`-Schnittstelle eigene Restriktionen an die Einbettung oder Entfernung eines `SinglePlayer`-Objekts aus einem Account zu stellen.

Die `SinglePlayer`-Klasse leitet sich - genau wie die `TeamPlayer`-Klasse - von der Oberklasse `Player` ab. Daher soll diese zunächst genauer betrachtet werden. Diese besitzt vier neue und wichtige Attribute, deren Bedeutung für das Framework nun erläutert wird. `isOnline` gibt an, ob das entsprechende Objekt momentan online ist, das heißt, ob ein Nutzer dieses Objekt momentan geladen hat. Dadurch ist beispielsweise eine Überprüfung, ob ein bekannter Nutzer momentan spielt und somit spielintern kontaktiert werden kann, möglich. Mit `mode` sollen verschiedene Spielmodi ermöglicht werden, um „Bezahl“-Spieler von „Gratis“-Spielern unterscheiden zu können. Theoretisch könnte dieses Attribut auch der `User`-Klasse angehängt werden, jedoch gestattet die gewählte Implementierung `Player`-Objekte verschiedenster Modi einem Nutzer zuzuordnen, während die Alternative für diesen Anwendungsfall mehrere `User`-Objekte vorraussetzen würde. Da - abhängig vom Spieltyp - ein Spieler ein Inventar benötigt, wird ihm genau ein `Container`-Objekt zugeordnet, das Gegenstände aufnehmen kann. Bei der letzten wichtigen Eigenschaft - `currentstate` - handelt es sich um die ID des Zustands aus dem Automaten des Spiels, in dem sich der Spieler momentan befindet. Dadurch kann geprüft werden, welche Aktionen ein Spieler im Augenblick ausführen kann oder welche Bedingungen gelten müssen, damit er in den nächsten Zustand gelangt. Nicht implementiert werden konnte die Berechnung des Punktestandes, da dieser zu sehr von dem Spiel abhängt. Daher wurde ein Interface geschaffen, mit dem dieser berechnet und bei Bedarf veröffentlicht werden kann.

Die beiden von `Player` abgeleiteten Klassen `SinglePlayer` und `TeamPlayer` fügen jeweils weitere Attribute hinzu. So können Einzelspieler andere Einzelspieler als Freunde definieren. Dadurch kann beispielsweise sichergestellt werden, dass unbekannte Personen, die nicht in der Freundesliste enthalten sind, keinen Kontakt mit dem Spieler aufnehmen dürfen. Während Freunde nur lose verknüpft sind, ist es den Spielern auch gestattet, sich in festen Verbünden, den Teams, zusammen zu schließen. Ein Spieler kann entweder genau ein Team anführen (`leads`) oder Mitglied eines Teams sein (`member`). Dadurch wird verhindert, dass ein Spieler mehreren Gruppen angehört, was zu Interessenskonflikten führen könnte. Schließlich besitzt jeder Spieler auch Positionsdaten. Wie diese allerdings bezogen werden können, muss bei der Implementierung mittels der definierten Schnittstelle angegeben werden. Die weiteren Methoden dieses Interfaces regeln, ob ein Freund hinzugefügt oder gelöscht werden kann und ob einem Team beigetreten oder dieses verlassen werden darf. Die Klasse der Teams hingegen benötigt lediglich Attribute zur Speicherung des benötigten

Anführers und ein Set ihrer Mitglieder. Der Anführer wird dafür benötigt, um Entscheidungen für das gesamte Team zu treffen. Für eine Prüfung, ob ein Spieler einem Team als Mitglied oder Anführer beitreten oder dieses verlassen kann, wird analog zur Klasse der `SinglePlayer` nur mittels Interface implementiert.

Die bereits angesprochene Verwaltung der Spielgegenständen erfolgt mittels der Klasse der `Container`. Jeder (sammelbare) Gegenstand muss sich daher im Besitz eines Objektes dieser Klassen befinden. Die Schnittstelle dieser Klasse ermöglicht es den Umgang mit diesen `Items` zu regeln, indem definiert wird, ob ein Gegenstand aufgenommen, gelöscht oder mit einem anderen `Container` ausgetauscht werden kann. Dies kann von mehreren Faktoren abhängen. Möglich ist es, dass ein Spieler beispielsweise eine maximale Traglast - repräsentiert durch das Attribut `containersize` - besitzt, die zu keinem Zeitpunkt überschritten werden darf und daher mit dem aktuellen Füllegrad (`fillingdegree`) abgeglichen werden muss. Da sich das „Gewicht“ eines Gegenstandes aber - spielabhängig - stak unterscheiden kann (maximale Anzahl je Gegenstand, maximale Anzahl unterschiedlicher Gegenstände, usw.), müssen bei der Implementierung Routinen definiert werden, die, vor jeder Veränderung der Gegenstände des `Containers`, den danach gültigen Füllegrad ermitteln. Durch diese Art der Verwaltung ergibt sich, dass auch die Spielwelt ein Spieler sein muss, da kein Gegenstand ohne Besitzer existieren kann. Dies ist jedoch keine wirkliche Einschränkung und kann beispielsweise durch Spieler eines `ADMINISTRATOR`-Accounts mit festen Positionsdaten realisiert werden.

Die letzte wichtige Klasse für mobile, ortsbasierte Spiele stellen die geografischen Objekte dar. Diese besitzen ebenfalls eine (Start-)Position, die sich allerdings - abhängig vom Objekttyp - ändern kann, wenn es nicht stationär (`isStatic`) ist. Wie diese Bewegung allerdings zu erfolgen hat (Verfolgung fester Pfade, KI-gesteuert), muss bei der Implementierung angegeben werden (`performeMove`). Zusätzlich kann angegeben werden, was bei einem Zusammenprall mit einem Spieler zu erfolgen hat (`onContact`). Dies könnte davon abhängen, ob ein betreffendes Objekt blockierende Eingenschaften besitzt, was bereits in den Attributen des `GObjects` (`isSolid`) eingestellt werden kann. Da es sich bei diesen Objekten nicht zwangsläufig um Repräsentationen realer, sichtbarer Objekte handelt, müssen sie dem Spieler auch auf seinem Display angezeigt werden können. Soll dafür nicht ein Standardsymbol zum Einsatz kommen, kann für jedes Objekt ein Verweis auf ein zu verwendendes Bild gesetzt werden (`iconurl`). Von der Oberklasse der `GObjects` lassen sich spezielle Ausprägungsformen ableiten. Im Framework ist dies bereits für kreisförmige und rechteckige Objekte erfolgt, die zusätzliche Informationen, wie ihren Radius oder die Position ihrer Eckpunkte, sowie Methoden, die ermitteln können, ob ein anderes Objekt in sie eingedrungen ist, bereitstellen.

Selbstverständlich leiten sind auch Spielgegenstände (`Item`) von dieser Klasse ab. Dabei handelt es sich um punktförmige Objekte, die als zusätzliches Attribut eine Mengenangabe beinhalten, um angeben zu können, wie viele Gegenstände dieses Typs sich an dieser Stelle befinden und - falls dies gestattet ist (`isCollectable`) - eingesammelt werden können. Die Bestimmung des Gegenstandwerts in der Spielwährung wird, ebenso wie die Regeneration der Objekte (`accretion`), durch eine Schnittstellenlösung realisiert, um dynamisch bestimmte Kurswerte unterstützen zu können.

Die Klasse des endlichen Automatens (`FSM`) wurde, wie in Abbildung 5.6 zu sehen, gemäß der herkömmlichen Definition eines FSM implementiert, weshalb für eine weiterführende Beschreibung auf die Literatur[4] verwiesen wird.

5.3.2. Die Klassen des `dao`- und `hibernate`-Pakets

Wie bereits bei der Beschreibung der Pakete erläutert, befinden sich im `dao`- und `hibernate`-Paket alle nötigen Methoden, um die Objekte des Frameworks persistent halten zu können. Daher wird für

[4] siehe [Scho8]

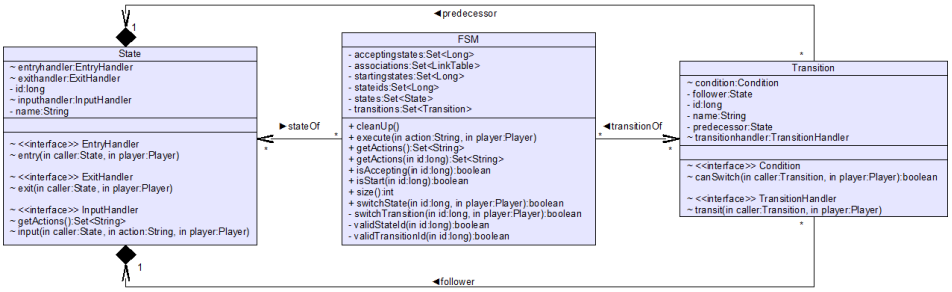

Abbildung 5.6.: Die Klassen des `domain`-Pakets - der endliche Automat

jeden Objekttyp ein eigenes Interface und dessen Implementierung benötigt. In ihrem Aufbau und ihrer Arbeitsweise unterscheiden sich diese Klassen allerdings - von den erwarteten Parametertypen abgesehen - praktisch nicht. Daher soll im Folgenden nur das `IUserDAO`-Interface betrachtet werden und für die restlichen Typen wird auf den Quellcode verwiesen.

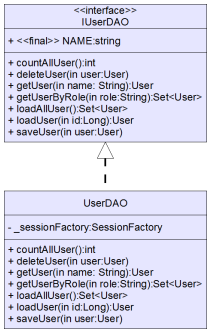

Abbildung 5.7.: Das IUserDAO-Interface und dessen Implementierung für Hibernate

In der Abbildung 5.7 ist das Klassendiagramm dieser beiden Klassen zu sehen. Von den praktisch selbstbeschreibenden Methoden finden sich `countAllXXX` (zur Ermittlung der Anzahl aller Objekte des Typs in der Datenbank), `deleteXXX`, `loadXXX`, `loadAllXXX` und `saveXXX` in den Interfaces für jede Klassen wieder, während die Methode `getUserByRole` nur für die User-Klasse (für die Suche nach Administratoren, normalen Nutzern und Gästen) und die Methode `getXXX` nur für Klassen mit benannten Objekten existieren.

5.3.3. Die Klassen des `gwt`-Pakets

Im `gwt`-Paket befinden sich die Implementierungen der vier für den Datenaustausch zwischen Client und Server verantwortlichen Schnittstellen. In der Regel werden die Anfragen allerdings nur an die entsprechenden Methoden des `service`-Pakets weitergeleitet. In Abbildung 5.8 ist ein verkürztes UML-Diagramm dieser unzusammenhängenden Klassen zu sehen.

Die Klasse `ConfigurationRemoteImpl` stellt dabei Methoden für die Initialisierung des Servers und für den Umgang mit einem (unerwarteten) Spielabbruch und mit timergesteuerten Ereignissen zur Verfügung. Die `MailRemoteImpl`-Klasse übernimmt keine spieltypischen Funktionen, sondern leitet

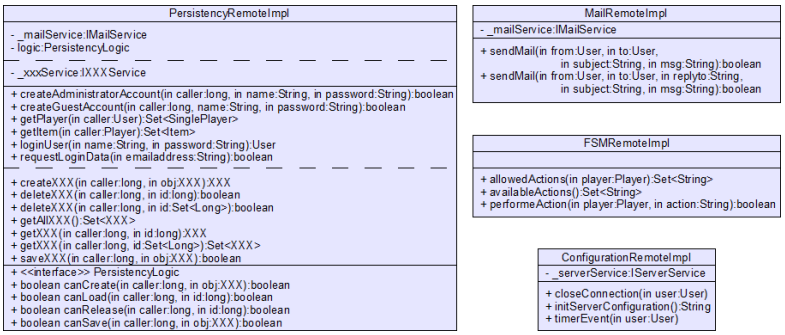

Abbildung 5.8.: Die Klassen des `gwt`-Pakets

die ihr übergebenen Parametern an die Services weiter, sofern es dem angegebenen Nutzer erlaubt ist, diese Mail abzuschicken. Da es in vielen Spielen gewünscht wird, den Spieler jederzeit über die im Augenblick möglichen Handlungen zu informieren, muss es dem Client möglich sein, auf den Zustandsautomaten Zugriff zu erlangen. Dies ermöglicht die Klasse `FSMRemoteImpl`. Durch sie können alle definierten und alle, dem Spieler in seiner gegenwärtigen Situation zur Verfügung stehenden, Aktionen abgerufen und ausgeführt werden.

Eine weitere wichtige Aufgabe, die einen Datenaustausch zwischen Client und Server benötigt, besteht darin, alle vom Framework angebotenen Objektypen persistent halten zu können. In der Klasse `PersistencyRemoteImpl` werden dafür Methoden bereitgestellt. Im Klassendiagramm wurden allerdings nicht alle davon modelliert, da diese nahezu vollständig redundante Informationen die Übersichtlichkeit stören würden und zum Verständnis der Arbeitsweise nicht benötigt werden. Daher wurde exemplarisch jeweils nur eine Methode mit dem Klassentyp `XXX` in das Diagramm eingetragen, wobei `XXX` für eine der in Abbildung 5.5 vorgestellten Klassen steht. Diese Methoden gestatten es einem `User`-Objekt eine Instanz einer dieser Klassen neu zu erstellen, zu löschen, zu laden und zu speichern. Ob dies dem Nutzer allerdings gestattet wird, kann mittels Interface je nach Anforderungen festgelegt werden. Außerdem können auch alle gespeicherten Objekte eines Typs abgerufen werden, um es beispielsweise einem Spielfeldeditor zu ermöglichen, alle definierten Gegenstände zu ermitteln, damit diese in einem Auswahlfeld angezeigt werden können.

Die restlichen (klassenspezifischen) Methoden sind selbsterklärend. Dennoch sollte kurz auf die `getPlayer`- und `getItem`-Methode eingegangen werden, um eventuelle Missverständnisse zu vermeiden. Im Gegensatz zu ihrem `getXXX` Pendant geben die `getPlayer`- und `getItem`-Methode alle `Player`-Objekte - beziehungsweise alle Gegenstände -, die sich im Besitz des Aufrufers befinden, zurück, während `getXXX` nur ausgewählte Objekte aus der Datenbank abruft.

5.3.4. Die Klassen der `services`- und `implementation`-Paketen

Da in den Schnittstellen und Implementierungen, die sich in den `services`- und `implementation`-Paketen befinden, nicht nur alle an die `gwt`-Services gerichteten Anfragen bearbeiten müssen, sondern auch große Teile der vom Framework vorgegebenen Spiellogik umsetzen, ist die Anzahl der hier realisierten Methoden sehr groß. Daher wurden die Methoden nach ihrem Aufgabengebiet sortiert und in eigene Klassen für jede existierende Spielklasse gekapselt. Zusätzlich zu diesen vier (Ober-) Klassen wird auch für die Initialisierung des Servers und für den Mailversand ein Service bereitgestellt. Abbildung 5.9 zeigt diese sechs so entstandenen Klassen.

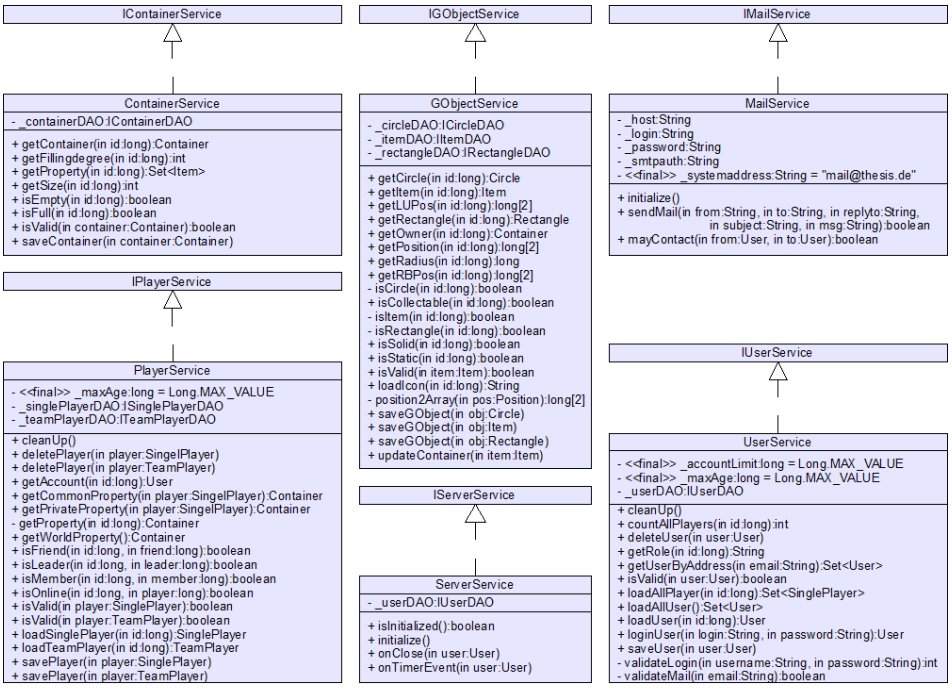

Abbildung 5.9.: Die Klassen des `implementation`-Pakets

Da sich die Funktionsweisen der hier befindlichen Methoden aus deren Namem erschließen lassen und diese meist nur dazu dienen, die aus der Datenbank erhaltenen Informationen zu filtern, wird im Folgenden nur auf einige ausgewählte näher eingegangen werden.

Die `ServerServices` beschäftigen sich mit der Initialisierung des Servers. In der beispielhaften Implementierung wird dafür zunächst geprüft, ob der Server bereits initialisiert wurde - in diesem Fall, ob ein `GUEST`-Account in der Datenbank existiert - und im Fehlerfall eine Instanz eines `GUEST-Users` angelegt. Da die beiden Methoden `onClose`, als Reaktion auf die Schließung des aktiven Browserfensters, und `onTimerEvent` zwar sehr wichtig für Spiele sein können, aber stark von dem Spielkonzept abhängen, erzeugen diese im Framework nur eine Systemausgabe, können aber dank der Architektur als Interface problemlos angepasst werden.

Die `MailService`-Klasse liest mit `initialize()` die Werte aus einer XML-Konfigurationsdatei, in der beispielsweise der SMTP-Host oder die benötigten Logindaten des Mail-Accounts gespeichert werden, aus. Ist diese Konfiguration erfolgt, kann mit `sendMail` aus den übergebenen Parametern ein gültiges `MimeMessage`-Objekt erstellt werden, das mittels der Methoden der JavaMail API versendet werden kann. Auch an dieser Stelle findet sich eine Möglichkeit, um zu überprüfen, ob ein Nutzer eine Mail an einen anderen Spieler senden darf.

In den restlichen Interfaces finden sich in erster Linie Anfragen an die Datenbank - beziehungsweise an die DAO-Schicht -, um Instanzen der zugehörigen Klassen zu laden, zu speichern oder upzudaten. Allerdings ist es auch möglich, nur Teilaspekte dieser Objekte abzurufen oder einfache Aggregationen auf diese anzuwenden. Erwähnenswert sind daher aus den User- und Player-Services nur die folgenden sieben Methoden:

`cleanUp`: Mit dieser Methode lassen sich Nutzer- und Spieler-Accounts automatisch löschen, wenn der letzte angegebene Login von diesen länger als `_maxAge` zurückliegt.

`getCommonProperty`: Hiermit kann der gemeinsame Besitz des Teams ermittelt werden, wenn der anfragende Spieler Anführer oder Mitglied dieses Teams ist.

`getPrivateProperty`: Diese Methode liefert den privaten Besitz eines Einzelspielers zurück.

`getWorldProperty`: Die Objekte der Spielwelt gehören Spielern, die einem `ADMINISTRATOR-User` zugeordnet sind. `getWorldProperty` liefert ein Set aus diesen Gegenständen zurück.

`isValid`: Da beispielsweise bei der Speicherung von Objekten diese vom Client übermittelt werden müssen, kann eine unsachgemäße Handhabung des Frameworks dazu führen, dass diese Objekte nicht die erwartete Form aufweisen. So können Attribute nicht belegt worden sein, die aber in der Datenbank als `NOT NULL` implementiert wurden. `isValid` prüft daher, ob alle empfangenen Daten gemäß der Definition wohlgeformt sind.

`validateLogin` **und** `validateMail`: Mit diesen Methoden lassen sich Logindaten und Emailadressen gemäß eines vorgegebenen Musters auf ihrer Validität hin überprüfen und - bei einem Verstoß gegen diese - eine entsprechende Fehlerbehandlung auszulösen.

Die `ContainerServices` und `GObjectServices` werden als vollständig selbsterklärend angesehen und bedürfen daher keiner weiteren Erläuterung.

5.3.5. Die Klassen des `support`-Pakets

Abbildung 5.10 zeigt die Klassen des `support`-Pakets. Bei der `GeoDataFunctions`-Klasse handelt es sich um eine Sammlung von Berechnungsfunktionen für geografische Daten. Diese Funktionen wurden statisch implementiert, da `GeoDataFunctions` keine Klasse im objektorientierten Verständnis darstellt und eine Instantiierung keinen Sinn ergeben würde. Für das Framework wurden lediglich zwei solche Funktionen implementiert: `getPosition` zur Umrechnung von der internen Repräsentation der Positionsdaten und der von GoogleMaps benötigten Darstellung und `getDistance` zur Abstandbestimmung zweier geografischer Objekte. Weitere Funktionen, wie Koordinatentransformationen sollten bei Bedarf daher dieser Klasse über das Interface `TransformationLogic` hinzugefügt werden.

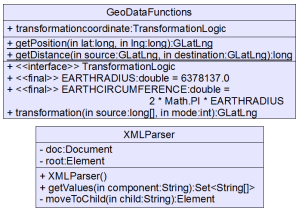

Abbildung 5.10.: Die Klassen des `support`-Pakets

Der ebenfalls in dieser Klasse implementierte XML-Parser verwendet den Xerces Java Parser, um Werte aus einer Konfigurationdatei, die valide zu der Schema-Definition aus Anhang C sind, auszulesen. Im Framework wird dies beispielhaft für die Konfiguration der Mailschnittstelle eingesetzt, allerdings sollte über die Ausweitung dieses Konfigurationssystem in späteren Versionen nachgedacht werden.

5.4. Die Implementierung

Die Implementierung der Klassen erfolgte nahezu ausschließlich in Java. Unterstützung erfuhr die Entwicklung dabei durch das gwt, unter anderem bei der Entwicklung der Oberflächen für die Widgets und bei den RPCs. Eine komplette Übersicht über die genutzten Java Bibliotheken befindet sich in Anhang A. Besonders erwähnenswert dabei ist der Einsatz von Hibernate und Gilead, für die persistente Datenhaltung, für dessen Mappingdaten Grundkenntnisse in XML vorausgesetzt wurden. Diese konnten ebenfalls für die Erstellung der Konfigurationsdatei des Frameworks und des Fragenkatalogs der Flash-Anwendung genutzt werden. Für die beispielhafte (und sehr minimalistische) Gestaltung der Oberflächen mussten darüberhinaus Kenntnisse über CSS und HTML erlangt werden. Bei der Implementierung des Quizzes wurde Flash eingesetzt, was neben einer Einarbeitung in die Flash-Umgebung auch die Programmierung von ActionScript 3 Code notwendig machte.

Tabelle 5.1 fasst den Aufwand, der für die einzelnen Komponenten des Frameworks erforderlich war, zusammen. Als Maßzahlen dienen dafür, neben der in etwa benötigten reinen Arbeitszeit[5], auch die physischen und logischen Zeilen an Quellcode (SLOC[6])[7]. Dadurch soll dem Leser ein Eindruck davon vermittelt werden, wie sich der benötigte Aufwand bei der Einarbeitung und Architektur zu dem der Implementierung verhält, und wo die Schwerpunkte dieser Arbeit liegen.

Komponente	**Sprache**	**SLOC**		**Arbeitszeit**
		physisch	**logisch**	
Clientseitige RPC Interfaces	Java	415	252	1 h
DAOs	Java	654	513	2,5 h
Exceptions	Java	181	136	0,5 h
Flash-Anwendung(*)	Flash, AS 3 und XML	622(*)	333(*)	50 h
Gestaltung und Layout	HTML und CSS	396	312	2 h
gwt-Schicht	Java	858	735	50 h
Initialliesierung von Gilead	Java	37	29	2,5 h
Klassendefinitionen	Java	2455	1951	200 h
Konfigurationsdateien	XML	299	270	50 h
Services	Java	1007	790	75 h
Support-Funktionen	Java	124	102	2,5 h
Widgets	Java	1670	1432	75 h
Summe		**8718**	**6855**	**511 h**

Tabelle 5.1.: Arbeitsaufwand bei der Implementierung

5.5. Das Flash-Quiz

Die Beschreibung des Frameworks soll aber nicht beendet werden, ohne dem Leser wenigstens einen kurzen Einblick in die ebenfalls entwickelte und dem Framework zugehörende Flash-Anwendung zu

[5] inklusive Einarbeitungs- und Planungsphase
[6] berechnet mit SLOC Metrics von http://microguru.com/sloc/
[7] Bei Flash-Programmen entsteht zusätzlicher Code in den Autorendateien, der nicht messbar ist. Daher verfälschte Ergebnisse bei (*)

verschaffen. Abbildung 5.11 zeigt die Oberfläche des Quizzes. Nachdem ein Fragenkatalog aus einer beiliegenden XML-Datei geladen wurde, wird im oberen Fensterbereich eine Frage und darunter vier[8] mögliche Antworten angezeigt. Wurde eine oder mehrere[9] Antworten ausgewählt, so werden diese anhand der XML-Datei ausgewertet.

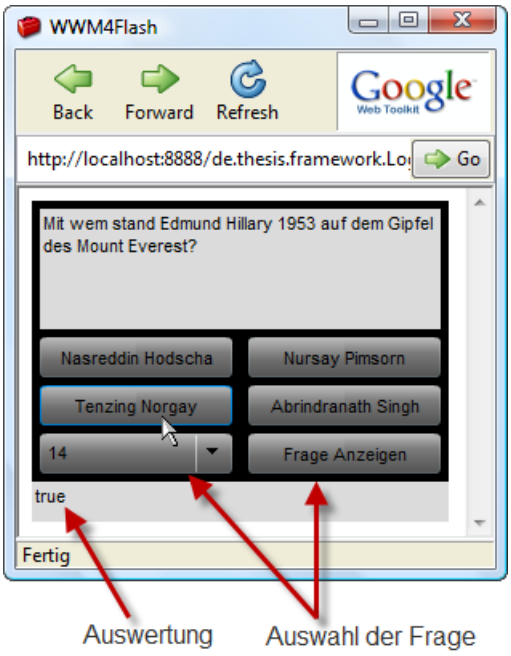

Abbildung 5.11.: Das Flash-Quiz

Der Prototyp enthält zwei weitere Komponenten. Zum einen können die Fragen eines Fragenkatalogs von Hand ausgewählt und einzeln geladen werden, ohne dass dafür eine Kommunikation mit der aufrufenden Clientanwendung stattfinden muss. Zum anderen wird dem Spieler die Auswertung direkt in einer Statusleiste angezeigt, so dass auch hierfür keine Kommunikation zwischen Flash und gwt erfolgen muss.[10]

[8] Das Schema erlaubt je Frage zwischen eine bis vier Antwortmöglichkeiten, der vorliegende Prototyp wurde aber auf genau vier beschränkt.
[9] Der Prototyp unterstütz lediglich Single-Choice-Fragen.
[10] siehe Kapitel 7, Punkt NFA-3

Kapitel 6

Der Prototyp

Nachdem in Kapitel 5 die Komponenten des erstellten Frameworks, das die Entwicklung von mobilen ortsbasierten Browserspielen unterstützen soll, vorgestellt wurden, wird nun ein einfacher Prototyp beschrieben, der im Rahmen dieser Arbeit entwickelt wurde und auf diesem Framework basiert. Dazu wird zunächst kurz auf die Grundidee des Spiels eingegangen, anschließend die konkrete Umsetzung beschrieben, bevor mit Ideen zur Erweiterung des Prototyps das Kapitel abgeschlossen wird.

6.1. Tic Tac Toe

Bei dem Spiel „Tic Tac Toe“ handelt es sich um eines der vielleicht beliebtesten Pen-&-Paper-Spielen der Welt, dessen Geschichte sich bis in das antike Rom zurückverfolgen lässt. Dabei belegen zwei Spieler abwechselnd die Felder eines in drei Zeilen und drei Spalten eingeteilten Spielbrettes. Gewonnen hat der Spieler, der als erstes eine komplette Reihe, Spalte oder Diagonale mit seinen Markierungen belegt hat. Sollten alle neun Felder belegt worden sein, ohne dass einer der Spieler dieses Spielziel erreicht hat, so wird die Partie als unentschieden gewertet. In Abbildung 6.1 ist ein dafür benötigtes Spielfeld dargestellt.

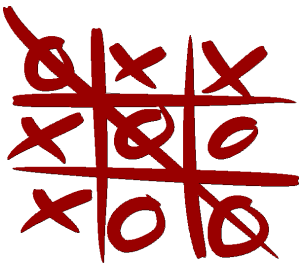

Abbildung 6.1.: Tic Tac Toe Spielfeld (Quelle: [Sym07])

Trotz (oder gerade wegen) dieses sehr einfachen und kurzen Spielprinzips, welches dafür sorgt, dass nach spätestens acht Zügen ein Ergebnis vorliegt, gibt es wohl nur wenige Menschen, die noch nie eine Partie Tic Tac Toe gespielt haben. Daher ist das Spiel auch Grundlage einiger mathematischer Studien geworden. Aus diesen ergibt sich allerdings, dass Tic Tac Toe ausgesprochen unfair ist. Geht

man von zwei gleichermaßen taktisch agierenden Spielern aus, so wird Spieler 1 rund 51% aller Spiele für sich entscheiden können, während Spieler 2 in maximal 31% der Fälle gewinnt. [Doo5]

Das überschaubare Regelwerk, die einfachen grafischen Voraussetzungen und die geringe Anzahl an Benutzerinteraktionsmöglichkeiten machten es möglich, dass Tic Tac Toe bereits 1952 als erstes grafisches Computerspiel der Welt für den EDSAC erscheinen konnte [Wino3]. Aus diesen Gründen wurde es auch als Testspiel für das Framework gewählt.

6.2. Pervasive Tic Tac Toe

„Pervasive Tic Tac Toe“ übernimmt große Teile des originalen Spielprinzips, verlagert das Spielfeld aber weg von den stationären Computern direkt in die Umgebung der Spieler. Des Weiteren wurde versucht, durch eine Anpassung des Regelwerks den Vorteil des Spielers, der den ersten Zug ausführen darf, abzuschwächen. Dieses neue, erweiterte Regelwerk wird in Abschnitt 6.2.1 genauer erläutert, während sich der Abschnitt 6.2.2 mit der eigentlichen Umsetzung auseinandersetzt.

6.2.1. Spielprinzip

Bei Pervasive Tic Tac Toe treten zwei Spieler gegeneinander an und müssen versuchen, auf einem 3 x 3-Felder großem Spielbereich eine vollständige Reihe, Spalte oder Diagonale zu besetzen. Dafür steht ihnen eine unbegrenzte Menge an Markern zur Verfügung. Im Gegensatz zu dem in Abschnitt 6.1 beschriebenen Spiel befindet sich dieses Spielfeld allerdings nicht auf einem Blatt Papier, sondern wird direkt auf die Umgebung des Spielers abgebildet.

Abbildung 6.2.: Oberfläche von Pervasive Tic Tac Toe

Damit die Spieler die Spielfelder sehen und damit interagieren können, benötigt daher jeder der beiden Kontrahenten ein mobiles Gerät mit einem Display, einem Internetanschluss und einem GPS-Empfänger. Nach der erfolgreichen Anmeldung auf dem Server wird ein passender Spielort mit einer vorgegebenen Größe gewählt und mit Zusatzinformationen, wie beispielsweise der genauen Anordnung der Felder oder der letzten bekannten Position des Gegners, angereichert auf den Displays der Spieler angezeigt, wie es in Abbildung 6.2 zu sehen ist. Die neun Felder sind dabei quadratisch und haben die gleiche Größe. In der Mitte jedes Feldes befindet sich ein sogenannter Hot Spot, der für die Belegung entscheidend ist. Die Startpositionen der Spieler befinden sich in den entgegengesetzten Eckpunkten im Nordwesten und Südosten. Sobald sie sich dort eingefunden haben, beginnt das Spiel. Spieler 1 darf ein freies Feld wählen und per Doppelklick aktivieren. Er belegt das Feld allerdings noch nicht, sondern schaltet es lediglich frei. Nun müssen beide Spieler versuchen, möglichst schnell zu dem Hot Spot dieses Feldes zu gelangen, da das Feld von dem Spieler belegt wird, der den Punkt bis auf einen vorgegebenen Abstand als erstes erreicht. Dadurch wird der Vorteil des ersten Zuges in so weit abgeschwächt, als dass dieser nicht gleichbedeutend mit der Eroberung des ersten Feldes sein muss[1].

Der Toleranzbereich um den Hot Spot muss gewährt werden, da sich unbegehbare Objekte auf diesem Platz befinden könnten und ein Spiel in dicht besiedelten Gebieten unmöglich machen würde. Jedoch gerade in stark bebauten Umgebungen steckt der Reiz an dem Spiel, da unerwartete Hindernisse - beispielsweise Mauern - das Erreichen eines Feldes erschweren und somit neben der Entfernung eines Spielers zum nächsten Feld und seine körperliche Fitness ebenfalls bei einer erfolgversprechenden Strategie berücksichtigt werden müssen.

Wurde ein Feld belegt, so wird dies auf der Karte der beiden Spieler angezeigt. Auch die Position des Gegners wird erst jetzt aktualisiert. Jedoch darf sich jeder Spieler nach der Belegung - unsichtbar für den Kontrahenten - in eine günstigere Position bringen. Im Anschluss ist der nächste Spieler mit der Aktivierung eines Feldes an der Reihe, bis entweder eine Gewinnsituation eintritt oder alle Felder belegt wurden. Daraufhin kann eine weitere Partie begonnen werden, sobald die Spieler sich erneut in ihre Ausgangsposition begeben haben.

6.2.2. Umsetzung

Bevor auf die wichtigsten Aspekte der Implementierung von Pervasive Tic Tac Toe näher eingegangen werden kann, sollte zunächst geklärt werden, welche Aktivitäten in dem Spiel möglich sein sollten und wer diese ausführen darf. In Abbildung 6.3 wird daher ein Aktivitätsdiagramm abgebildet. Dabei wird zwischen clientseitigen und serverseitigen Aktivitäten unterschieden, wobei erstere eine Aktion des Spielers erwarten.

Eine Partie Pervasive Tic Tac Toe beginnt stets damit, dass sich (mindestens) zwei Spieler an dem Server anmelden oder registrieren. Einem `User` wird dabei eine `SinglePlayer`-Instanz fest zugewiesen, da für dieses Spielprinzip weder mehrere Accounts noch Teams benötigt werden. Daher kann das `LoginManager`-Widget aus dem Framework übernommen werden, wobei eine erfolgreiche Anmeldung oder Registrierung direkt zu einem Spieleinstieg führen sollte. Daraufhin sucht der Server einen angemeldeten, spielbereiten Gegenspieler heraus, der sich, laut seiner letzten übermittelten Position, möglichst nahe bei dem Spieler befindet. War dies möglich, wird für dieses Spielerpaar ein `SinglePlayer`-Objekt, das sich im Besitz eines `ADMINISTRATOR`s befindet und neun Marker „besitzt“, sowie ein Spielfeld erstellt, dessen Zentrum mittig zwischen den Beiden gesetzt wird. Den Spielern wird neben der Spiel-Id, die mit der des `ADMINISTRATOR`-Spielers übereinstimmt, auch eine Startposition mitgeteilt, zu der sie sich begeben müssen, bevor das eigentliche Spiel beginnen kann.

[1] Dadurch werden die Felder nicht zwangsläufig gleichmäßig unter den Spielern aufgeteilt!

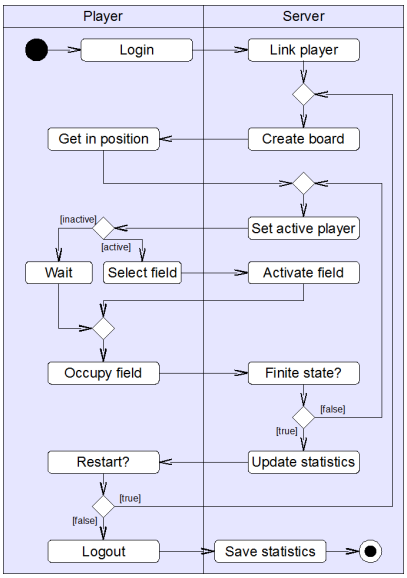

Abbildung 6.3.: Aktivitätsdiagramm von Pervasive Tic Tac Toe

Der Server muss in der Folge bestimmen, wer den ersten Zug ausführen darf. Für die erste Partie geschieht dies zufällig, danach wird abwechselnd begonnen. Der aktive Spieler - festgehalten als neues Attribut der `SinglePlayer`-Klasse - wählt daraufhin ein Spielfeld aus, während sein Kontrahent auf die Freischaltung warten muss. Diese erfolgt durch den Server, wenn das gewählte Feld aktivierbar, das heißt noch nicht belegt, ist. Für die Aktivierung wird im Zentrum des Feldes ein `Circle`-Objekt mit einem Radius, der mit dem festgelegten Toleranzbereich übereinstimmt, erzeugt. Die `id` eines beliebigen Markers aus dem Besitz des `ADMINISTRATOR`-Spielers wird an die Clients übermittelt, was gleichzeitig die Freischaltung bedeutet. Die Spieler versuchen, diesen Gegenstand möglichst schnell einzusammeln. Dies wird durch ein Tauschgeschäft mit dem `ADMINISTRATOR` realisiert, das genau dann zustande kommt (`canExchange == true`), wenn es sich noch im Besitz des Computers befindet und der Spieler sich innerhalb des aktivierten Toleranzbereichs aufhält.

Die restlichen Methoden der `ContainerLogic`-Schnittstelle können fest definierte Standardwerte - `calculateXXXFillingDegree` $\rightarrow$ `0`, `canAdd` $\rightarrow$ `false`, wenn Spieler == Admin, `true` sonst und `canRemove` $\rightarrow$ `true`, wenn Spieler == Admin, `false` sonst - liefern.

Im Anschluss an einen erfolgreichen Tausch muss der Server überprüfen, ob sich dadurch für den Spieler eine Gewinnsituation ergeben hat oder ob alle Marken verteilt wurden und damit ein Unentschieden eingetreten ist. Ist beides nicht der Fall, muss der nächste Spieler darüber benachrichtigt werden, dass er am Zug ist. Dies wird so lange wiederholt, bis eines der beiden Ereignisse eintritt, was zu einer Aktualisierung der Statistiken führt. Daraufhin können die beiden Spieler entscheiden, ob sie in dieser Paarung noch ein Spiel starten wollen, sie sich ausloggen, um einen neuen Partner zu finden, oder das Spiel vollständig beenden möchten. Letzteres führt unweigerlich zu einer Speicherung der Spielerdaten und löscht alle temporär benötigten Daten des Servers, wie beispielsweise den Spieler des Administrators. Nicht im Aktivitätendiagramm ersichtlich ist, dass auch ein Verbindungsabbruch - gemeldet von dem `WindowCloseListener` - dies bewirkt. Allerdings ist damit zusätzlich eine Niederlage im aktuellen Spiel als Strafe verbunden.

6.3. Variationsmöglichkeiten

Neben der in Abschnitt 6.2 vorgestellten Variante von Tic Tac Toe, die sich noch relativ nahe an dem Originalspiel orientiert, sollen noch vier weitere Möglichkeiten für eine Umsetzung als mobile ortsbasierte Browserspiele vorgestellt werden, die leider aus Zeitgründen nicht mehr implementiert werden konnten. Zum einen werden dadurch weitere Einsatzmöglichkeiten des Frameworks vorgeführt und zum anderen die Mächtigkeit der auf den ersten Blick einfachen und stark begrenzten Spiellogik hinter Tic Tac Toe demonstriert werden.

6.3.1. Pervasive Tic Tac Toe mit begrenzten Ressourcen

Da bei der klassischen Spielart durchschnittlich in jeder fünften Partie kein Sieger ermittelt werden kann, könnte durch eine Beschränkung der Markierungen, die jedem Spieler zur Verfügung gestellt werden, dieses Problem umgangen werden.

Die Variante „Pervasive Tic Tac Toe mit begrenzten Ressourcen“ verläuft grundsätzlich wie eine normale Partie Pervasive Tic Tac Toe. Zwei Spieler aktivieren abwechselnd eines von zunächst neun freien Feldern auf einer Landkarte und müssen danach versuchen, möglichst schnell den Hot Spot dieses Feldes zu erreichen, um das Feld einer Markierung zu besetzen. Allerdings stehen jedem Spieler dafür maximal vier Marker zur Verfügung. Hat ein Spieler alle Marken verbraucht und es konnte noch kein Sieger ermittelt werden, muss der Spieler nach der Aktivierung eines neuen Feldes zunächst in ein von ihm bereits besetztes Feld zurückkehren und seinen Spielstein von dort entfernen. Dadurch wird dieses Feld allerdings wieder frei und kann in der nächsten Runde erneut aktiviert und besetzt werden.

Neben der Tatsache, dass auf diese Weise auf jeden Fall ein Sieger ermittelt werden kann, da zu jedem Zeitpunkt maximal acht Felder belegt sein können, müssen die Spieler vollkommen neue Taktiken entwickeln, um das Spiel für sich entscheiden zu können. So macht es keinen Sinn mehr zu versuchen, im Wettlauf mit dem Kontrahenten jedes Feld als Erster zu erreichen und somit unter Umständen eine wertvolle Marke unnötig zu verbrauchen. Andererseits bleibt für diese Entscheidung nur sehr wenig Zeit, da ansonsten der Vorsprung des Gegenspielers möglicherweise bereits zu groß sein kann. Allerdings dauert eine einzelne Partie in dieser Variante vermutlich erheblich länger, da sie in den seltensten Fällen nach acht Zügen beendet sein wird.

Die für diese Variante benötigte Ressourcenverwaltung kann stark vom Framework unterstützt werden. Das ursprüngliche Design kann nahezu unverändert von Pervasive Tic Tac Toe übernommen werden. Lediglich die Anzahl der Items, die an die beiden Spielern zu Spielbeginn ausgegeben werden, muss von neun auf vier herabgesetzt werden. Außerdem muss nun auch der Austausch einer Marke von einem Spielfeld an einen Spieler ermöglicht werden. Dies lässt sich aber durch einfache `if`-Abfragen - beispielsweise ob die Farbe der Marke mit der des Spielers übereinstimmt - bewerkstelligen. Die Containergröße der Spieler kann zwar ebenfalls auf vier reduziert werden, dies ist allerdings nicht zwingend erforderlich, da bereits die Anzahl der Marken begrenzt ist und keine weiteren erzeugt werden können. Um die Spielidee aber möglichst exakt abzubilden, sollte diese Anpassung dennoch bei einer Implementierung in Betracht gezogen werden.

6.3.2. Pervasive Tic Tac Toe in Teams

Während die bekannten Versionen stets auf klassische Eins-Gegen-Eins-Duelle setzen, so eignet sich gerade die ortsbasierte Spielart auch für Mehrspielerspiele in zwei Gruppen. Bei „Pervasive Tic Tac Toe in Teams“ treten zwei Viererteams gegeneinander an. Jedes Teammitglied steht dabei für eine

Marke. Eines der Teammitglieder übernimmt die Rolle des Anführers, dessen Aufgabe darin besteht, mit dem Server zu interagieren. Haben sich die Teams in ihre Ausgangsposition begeben, so wird - genau wie bei der normalen Variante - von ihm ein Feld aktiviert. Besetzen kann das Feld daraufhin grundsätzlich jedes Teammitglied, aber es bleibt nur so lange besetzt, wie dieser Spieler in dem Hot Spot stehen bleibt. Verlässt er es, so geht der Zustand erneut in „frei“ über und es kann in der nächsten Runde erneut aktiviert werden. Auch hier gilt es, als erstes Team eine Reihe, Spalte oder Diagonale zu besetzen.

Die Gründe dennoch ein Feld aufzugeben können dabei unterschiedlich sein:

- Es kann sein, dass jedes Teammitglied bereits ein Feld besetzt hält, aber noch keine Gewinnsituation eingetreten ist.
- Ein Feld kann auf Grund einer neuen Spielsituation und veränderten Positionen der gegnerischen Spieler seinen taktischen Nutzen für ein Team verloren haben.
- Ein Feld wird aufgegeben, um eine gute Positionierung eines Teammitgliedes ausnutzen zu können, damit eine Niederlage vom Team abgewandt werden kann.
- Schließlich kann es auch durch schlechte teaminterne Absprachen zu einer versehentlichen Aufgabe eines Feldes kommen.

Wie man anhand dieser neuen taktischen Möglichkeiten erkennen kann, ist eine gute Absprache innerhalb des Teams - sowohl vor einer Aktivierung als Berater des Teamleiters, als auch im Anschluss bei der Wahl, wem die Aufgabe des Läufers zukommen soll - in dieser Variante unumgänglich. Während die Aufgaben der Teamverwaltung noch größtenteils direkt von dem Framework unterstützt werden, so mangelt es an einer für diese Situation angepasste Kommunikationskomponente.

Diese Ermangelung kann aber auch als Teil des Spiels eingebunden werden, indem es die Aufgabe eines Spielers ist, Informationen zwischen den restlichen Mitgliedern auszutauschen, was auf Grund der räumlichen Nähe möglich sein sollte. Dieses Problem könnte aber auch durch eine Erweiterung der kartographischen Schnittstelle, die es den Spielern gestattet, Zeichnungen in der Karte vorzunehmen, die nur für das eigene Team sichtbar werden. Eine weitere Möglichkeit wäre es, sprachgesteuerte Kommunikationsmöglichkeiten zu implementieren, was allerdings mit einem hohen Realisierungsaufwand verbunden wäre.

Die wohl einfachste Lösung des Kommunikationsproblems wäre es, überhaupt keine Kommunikationsmöglichkeiten anzubieten, sondern diese Aufgabe an die Spieler zu delegieren. Diese müssen dann, beispielsweise mittels Mobiltelefonen, den Kontakt zu ihren Teammitgliedern halten. Auch wenn dadurch weder beim Framework, noch bei der Implementierung eines Spiels ein zusätzlicher Aufwand entsteht, sollte man sich darüber bewusst sein, dass durch diese ausgelagerte Lösung die Kommunikationskanäle nicht vom System überwacht werden können. Problematisch ist dies, wenn das Spiel dafür Reglementierungen einführt - man stelle sich beispielsweise beschränkte Kommunikationszeiten vor. Betrugsversuche gegen diese Regel könnten nur von einem (neutralen) menschlichen Spielleiter festgestellt und geahnded werden, oder man muss auf das blinde Vertrauen der Mit- und Gegenspieler setzen.

6.3.3. Pervasive Tic Tac Toe mit Runner

„Pervasive Tic Tac Toe mit Runner“ lehnt sich stark an existierende Pervasive Games wie Can You See Me Now (Abschnitt 2.4.5) oder Uncle Roy All Around You (Abschnitt 2.4.12) an. Die zwei konkurrierenden Parteien, deren Ziel es ebenfalls ist, Marken gemäß der Regeln des originalen Spiels auf einem Spielfeld zu verteilen, bestehen in dieser Variante aus zwei Personen. Eine der

beiden bewegt sich in der realen Welt und muss versuchen, einen aktivierten Hot Spot vor ihrem gegnerischen Pendant zu erreichen, um somit das Feld für ihre Seite zu besetzen. In diesem Punkt unterscheidet sich der Spielaufbau der Variante mit Runner von der implementierten normalen Version nicht.

Erst der Einsatz des zweiten Spielers verändert das Spielgefühl völlig. Dieser verfolgt das Spiel nur vom heimischen Computer aus. Alles was er dabei sieht ist ein normales Tic Tac Toe Spielfeld, wie es beispielsweise in Abbildung 6.1 zu sehen ist. Auf diesem wählt er abwechselnd mit seinem Gegner Felder aus, um eine Dreierreihe zu besetzen. Er kann allerdings keine Marken setzen, sondern das jeweilige Feld nur aktivieren. Die eigentliche Besetzung kann nur durch die Runner erfolgen. Diesen wird auf ihrem Display nur eine Karte ohne jegliche weitere Informationen, wie der Lage der Spielfelder oder die bereits besetzten Bereiche, angezeigt. Erst wenn ein Feld aktiviert wurde, wird die Lage seines Hot Spots eingeblendet und kann daraufhin besetzt werden.

Für ein erfolgreiches Spiel muss daher der Mitspieler über die ihm unbekannten Gegebenheiten aus der realen (z.B. Position des Mitspielers, Position des Gegenspielers, Lage von unumgänglichen Hindernissen, usw.) und virtuellen Welt (z.B. relative Lage des nächsten wichtigen Punktes schon vor der Aktivierung, usw.) in Kenntnis gesetzt werden. Erst mit Hilfe dieser Informationen, die sich durch die Kombination der beiden Spielwelten ergibt, kann eine sinnvolle Taktik entwickelt werden.

Der Implementierungsaufwand dieser Variante ist trotz Framework erheblich höher als bei den anderen Spielarten, da zwei Spielmodi mit jeweils vollkommen verschiedenartigen Oberflächen entwickelt werden müssen. Außerdem müssen zwei unterschiedliche Player-Objekte mit individuellen Fähigkeiten erstellt werden, was durch die Attribut-Eigenschaft und der Erweiterbarkeit der Klasse zwar möglich, aber auch sehr zeitintensiv ist. Die größte Schwachstelle des Frameworks für diese Variante stellen aber die Kommunikationskomponenten dar, da sich weder das Mailtool, noch ein Forum für die schnelle und direkte Kommunikation eignet. Ein herkömmlicher Chat würde ausreichen, aber die schnelle Texteingabe erweist sich auf mobilen Geräten meist als schwer bis unmöglich. Daher sollte für diese Variante ein Voice-Chat entwickelt werden.

6.3.4. Ortsversetztes Pervasive Tic Tac Toe

Während sowohl das implementierte Pervasive Tic Tac Toe, als auch die Variationen 6.3.1 - 6.3.3 davon ausgehen, dass sich die Spieler zu Spielbeginn in räumlicher Nähe aufhalten, so dass sie sich relativ schnell in die ermittelte Ausgangsposition begeben können, ist dies keinesfalls zwingend erforderlich. „Ortsversetztes Pervasive Tic Tac Toe“ setzt die echten Koordinaten, die von dem GPS-Signal geliefert werden, in Koordinaten einer internen virtuellen Karte um. Dabei wird die aktuelle Position von Spieler 1 automatisch als nordwestlichen Eckpunkt der internen Karte angesehen, während Spieler 2 den südöstlichen markiert. Die so entstandene Karte wird gleichmäßig in neun Felder mit jeweils einem Hot Spot eingeteilt. Wurden alle spielrelevanten Punkte berechnet, so werden diese an die Spieler übermittelt und relativ zu ihrem Aufenthaltsort in reale Positionsdaten überführt.

Damit ändert sich am Spielablauf nichts. Selbst Kombinationen mit den anderen vorgestellten Modi werden dadurch nicht verhindert. Auch für die Spieler ändert sich zunächst nur sehr wenig. Allerdings ist es weitaus weniger problematisch, einen Spielpartner zu finden, da dessen Aufenthaltsort vollkommen unentscheidend ist. Diese Freiheit wird aber durch kleinere Einschränkungen erkauft. So kann es vorkommen, dass die Wege zu den Hot Spots für einen Spieler schwer zugänglich sind, während sich der andere Spieler im freien Gelände befindet und dieser dadurch immer im Vorteil ist. Auch fördert der häufige Sichtkontakt die Motivation um jedes Feld zu kämpfen und ist somit für den Spielspaß zuträglich.

Dennoch würde dieser Spielmodus für viele Spieler vermutlich die einzige Möglichkeit darstellen eine Partie Pervasive Tic Tac Toe in Teams zu spielen, da es sehr schwer sein dürfte, acht Personen zur selben Zeit am selben Ort zu versammeln und mit der benötigten Hardware auszustatten. In diesem Fall lassen sich auch die genannten Restriktionen verkraften. Daher sollte ortsversetztes Pervasive Tic Tac Toe nicht als eigenständiges Spiel, sondern als eine Notlösung für dünnbesiedelte Gebiete angesehen werden.

Kapitel 7

Evaluation

Abschließend soll eine kritische Bewertung des entwickelten Frameworks erfolgen. Dabei müssen zwei unterschiedliche Aspekte betrachtet werden. Zum einen wird geprüft, in wie weit das Framework die gestellten Anforderungen aus Kapitel 4 erfüllt und zum anderen muss bewertet werden, welchen praktischen Nutzen es bei der Entwicklung von mobilen ortsbasierten Browserspielen haben kann. Letzteres basiert auf den bei der Implementierung von Pervasive Tic Tac Toe gewonnenen Erfahrungen.

- **FA-1:** Das Framework muss einen Zustandsautomaten simulieren können.

  Mit der Klasse FSM wird ein anpassbarer Zustandsautomat realisiert.

- **FA-2:** Das Framework muss neue Accounts anlegen können.

  FA-5: Das Framework muss einen Loginmechanismus bereitstellen.

  FA-6: Das Framework muss dem Spieler vergessene Passworte übermitteln können.

  FA-10: Das Framework muss Benutzerprofile anlegen können.

  Mit dem `LoginManager`-Widget wird eine GUI angeboten, die automatisch die entsprechenden Services zur Accounterstellung, dem Login und der Übermittlung vergessener Passwörter aufruft. Diese Funktionen können allerdings auch ohne den Einsatz des Widgets verwendet werden.

- **FA-3:** Das Framework muss fehlerhafte Accounts löschen können.

  FA-4: Das Framework muss Benutzerdaten verifizieren können.

  Der Server prüft für alle Objekte, bevor sie in die Datenbank übernommen werden, ob diese einem vorgegebenen Schema entsprechen. Da somit keine Fehlerhaften Accounts angelegt werden können, müssen diese auch nicht gelöscht werden.

- **FA-7:** Das Framework muss auf unerwartete Spielabbrüche reagieren können.

  Für Events des `WindowCloseListener` kann auf die Methode `onClose`, die von dem Paket `ServerService` bereitgestellt wird, zurückgegriffen werden.

- **FA-8:** Das Framework muss ein erweiterbares Benutzerprofil zur Verfügung stellen.

  Die `User`-Klasse besitzt das Attribut `attributes`, das beliebig viele neue Attribute speichern kann. Außerdem kann die Klasse bei Bedarf auch durch Vererbung um weitere Methoden und Attribute erweitert werden.

- **FA-9:** Das Framework muss ein anonymes Gastprofil erstellen.

  Bei der Initiallisierung des Servers wird von `initialize()` automatisch ein `GUEST`-Profil erstellt.

- **FA-11:** Das Framework muss Benutzerprofile verwalten können.

  FA-18: Das Framework muss statische Objekte verwalten können.

  FA-21: Das Framework muss erweiterbare Ressourcen verwalten können.

  Durch den Einsatz von Hibernate und den, in den Klassen des `dao`-Pakets definierten, Methoden ist eine persistente Datenverwaltung für alle Objekttypen gegeben.

- **FA-12:** Das Framework muss Benutzerprofile manipulieren können.

  FA-13: Das Framework muss eine Oberfläche zur Profilmanipulation bereitstellen.

  FA-14: Das Framework muss Benutzerprofile löschen können.

  Das `AdministrationManager`-Widget bietet eine GUI, um die Services zur Manipulation und dem gezielten Löschen von Profilen aufzurufen. Diese Services können jederzeit auch losgelöst von dem Widget verwendet werden.

- **FA-15:** Das Framework muss unterschiedliche Spielmodi verwalten können.

  Das `User`-Profil besitzt das Integer-Attribut `mode`, wodurch 4294967296 Modi unterschieden werden können.

- **FA-16:** Das Framework muss Benuterprofile miteinander verknüpfen können.

  Profile können nicht direkt, sondern nur über ihre `SinglePlayer`-Objekte (`friends` oder `leads` bzw. `member`) miteinander verknüpft werden.

- **FA-17:** Das Framework muss unbenutzte Benutzerprofile löschen können.

  Die Methode `cleanUp`, die für `User`- und für `Player`-Objekte existiert, löscht automatisch alle Karteileichen.

- **FA-19:** Das Framework muss Objekte darstellen können.

  Einem `GObject` kann ein beliebiges Icon zugeordnet werden, dessen Darstellung allerdings von der Implementierung der Spielorberfläche abhängt.

- **FA-20:** Das Framework muss Objekten Eigenschaften zuweisen können.

  Neben den, für die unterschiedlichen `GObject`-Unterklassen definierten Attributen, können beliebig viele weitere Attribute durch `attributes` angehängt werden.

- **FA-22:** Das Framework muss alle Ressourcentypen ermitteln können.

  Eine Methode `loadAllXXX` existiert für alle Objekttypen im `services`-Paket.

- **FA-23:** Das Framework muss eine allgemeine abstrakte „Währung“ einführen.

  FA-24: Das Framework muss eine Schnittstelle zur Implementierung von „Wechselkursen“ besitzen.

  Die Methode `calculateValue`, deren Implementierung dem Spielentwickler überlassen wird, liefert den aktuellen Wert einer Ressource. Intern wird dieser Wert als normale Integer-Variable verwaltet.

- **FA-25:** Das Framework muss die Ressourcen aus den Benutzerprofilen auslesen können.

  Der Besitz eines Spielers lässt sich über das Attribut `items` seinen zugehörigen `Container`-Objekts ermitteln.

- **FA-26:** Das Framework muss Ressourcen zwischen der virtuellen Welt und den Spielern austauschen können.

  FA-28: Das Framework muss Ressourcen unter den Spielern austauschen können.

  Inhalte von `Container`-Objekten können, sofern das Regelwerk dies gestattet (`canExcange`), unabhängig vom Besitzer der Ressource ausgetauscht werden.

- **FA-27:** Das Framework muss Ressourcen erschaffen und verbrauchen können.

  Die Methode `accretion` simmuliert eine Bestandsveränderung einer Ressource, durch einen positiven oder negativen Zuwachs.

- **FA-29:** Das Framework muss den Punktestand jedes Spielers in der Spielwährung ermitteln können.

  FA-30: Das Framework muss eine Highscoretabelle erstellen können.

  FA-31: Das Framework muss den Punktestand exportieren können.

  Über das Interface `PlayerLogic` kann der aktuelle Punktestand eines Spielers ermittelt (`calculateScore`) und exportiert (`submitScore`) werden, wenn dies gestattet ist (`maySubmit`). Alle Schnittstellen, die für die Erstellung einer Highscoretabelle benötigt werden, werden von dem Framework angeboten. Die Implementierung wird dem jeweiligen Spiel überlassen.

- **FA-32:** Das Framework muss die aktive Spielzeit der Spieler bestimmen können.

  FA-33: Das Framework muss die gesamte Spielzeit der Spieler bestimmen können.

  Das Framework kann die aktuelle (`getCurrentPlaytime`) und gesamte Spielzeit (`getPlaytime`) für `Player`-Objekte ermitteln.

- **FA-34:** Das Framework muss das „Alter“ der virtuellen Objekte bestimmen können.

  Für alle Objekte existiert die Methode `getAge`.

- **FA-35:** Das Framework muss die aktuelle Spielzeit bestimmen können.

  Mit der Klasse `java.util.Date` lässt sich das aktuelle Systemdatum ermitteln.

- **FA-36:** Das Framework muss mit timergesteuerten Ereignissen umgehen können.

  Auf timergesteuerte Ereignissen kann mit der Interface-Methode `onTimerEvent`, die von den `ServerServies` bereitgestellt wird, reagiert werden.

- **FA-37:** Das Framework muss Teams als Spielerobjekte verwalten können.

  FA-38: Das Framework muss Benutzerprofile von Teamspielern erweitern können.

  Die Klasse `TeamPlayer` leitet sich, genau wie die Klasse `SinglePlayer`, von der Klasse `Player`. Daher spielt es für den Umgang mit den Profilen keine Rolle, ob es sich dabei um Teams oder Einzelspieler handelt.

- **FA-39:** Das Framework muss eine Mitgliederverwaltung anbieten.

  Aus Zeitgründen konnte keine GUI zur Teamverwaltung implementiert werden, obwohl alle benötigten Methoden dafür existieren und die Entwicklung analog zu der des `PlayerManager`-Widgets erfolgen könnte.

- **FA-40:** Das Framework muss den aktuellen Spielzustand persistent halten können.
- **FA-41:** Das Framework muss alte Spielzustände wiederherstellen können.

  Der aktuelle Spielzustand wird im Spielerprofil unter dem Attribut `currentstate` hinterlegt. Für die Pesistenz ist Hibernate verantwortlich.
- **FA-42:** Das Framework muss kontextabhängig die Speicherung erlauben oder verbieten können.

  FA-43: Das Framework muss Benutzerdaten persistent halten können.

  FA-44: Das Framework muss Spieldaten persistent halten können.

  Alle definierten Objekte werden nur dann persistent gehalten, wenn das Regelwerk dies gestattet (`canSave`). Hibernate sorgt dann für die fehlerfreie Umsetzung.
- **FA-45:** Das Framework muss Daten geheim halten können.

  Alle Daten werden nur übermittelt, wenn der Anfragesteller dazu autorisiert ist (`canCreate`, `canLoad`, `canRelease` und `canSave` aus der Klasse der `PersistencyRemotImpl`).
- **FA-46:** Das Framework muss ein Forum anbieten.

  FA-47: Das Framework muss einen Chat anbieten.

  Aus Zeitgründen konnte weder ein Forum, noch ein Chat als Eigenentwicklung in das Framework eingebunden werden. Daher muss auf die zahlreichen Open Source Lösungen verwiesen werden (siehe Seite 93).
- **FA-48:** Das Framework muss Mailfunktionalitäten besitzen.

  Das Framework besitzt durch die `MailService`-Klasse einfache und leicht konfigurierbare Mailfunktionen.
- **FA-49:** Das Framework muss einen Editor anbieten.

  Es konnte kein grafischer Editor implementiert werden, obwohl alle dafür benötigten Methoden im Framework existieren, da eine (bedienbare) Oberfläche sehr viel Zeit in Anspruch nimmt.
- **FA-50:** Das Framework muss die Verfügbarkeit eines GPS-Signals ermitteln können.

  FA-52: Das Framework muss die Position des Spielers ermitteln können.

  Die Methode `getCurrentPos` aus dem Interface `SinglePlayerLogic` liefert, unabhängig von der eingesetzten Technik, die aktuelle Position eines `SinglePlayer`-Objekts. Für den Umgang mit Signalstörungen, ist der Spielentwickler verantwortlich.
- **FA-51:** Das Framework muss auf Positionsupdates reagieren können.

  Die Methode `updatePos` aktuallisiert die Position eines `SinglePlayer`-Objekts.
- **FA-53:** Das Framework muss die Position von Objekten der Spielwelt beziehen können.

  Für jedes `GObject` wird dessen Längen- und Breitengrad im Attribut `pos` gespeichert.
- **FA-54:** Das Framework muss den Abstand des Spielers zu den Objekten der Spielwelt ermitteln können.

  Die Methode `getDistance` aus der Klasse der `GeoDataFunctions` berechnet den Abstand zwischen zwei angegebenen Positionsdaten.
- **FA-55:** Das Framework muss weitere ortsabhängige Ereignisse definieren.

  Jedes definierte GObject kann durch die Methode `onContact` auf einen Spielerkontakt reagieren.

- **FA-56:** Das Framework muss Koordinaten unterschiedlicher Systeme umrechenen können.

  Auf Grund der großen Anzahl an unterschiedlichen Koordinatensystemen kann eine Koordinatentransformation nur als Schnittstelle der GeoDataFunctions-Klasse angeboten werden.

- **FA-57:** Das Framework muss ein erweiterbares Flash Quiz einbinden können.

  Es wurde ein Flash-Quiz erstellt, das mit jedem beliebigen XML-Fragenkatalog, der zu dem vorgegebenen XML-Schema valide ist, zusammenarbeitet.

- **NFA-1:** Das Framework darf die Einbindung externer Systeme zur Benutzeridentifikation nicht verhindern.

  Da der vollständige Login-Prozess als Interface mit einer einfachen Implementierung vorliegt, können durch einen Austausch dieser Implementierung auch externe Benutzeridentifikationsysteme in Kombination mit dem Framework genutzt werden.

- **NFA-2:** Das Framework darf das Einbinden von Multimediaformaten nicht verhindern.

  Jeder webfähige Medientyp kann mit dem Framework verwendet werden, sofern der verwendete Server und Browser mit dem Datenformat umgehen können.

- **NFA-3:** Das Framework darf die Kommunikation zwischen den Minispielen und dem Hauptspiel nicht verhindern.

  Flashanwendungen lassen sich direkt mit dem `FlashWidget` einbetten, aber ein Datenaustausch mit diesen ist zur Zeit nicht ohne Probleme möglich[1]. Allerdings kann die Flash-Anwendung von Hand in die HTML-Seite eingebunden und mittels JavaScript kontaktiert werden.

Wie man anhand der Auflistung sieht, werden nahezu alle herausgearbeiteten Anforderungen vollständig erfüllt. Die fehlenden Implementierungen lassen sich relativ einfach - allerdings verbunden mit einem hohen Zeitaufwand - nachträglich einbringen. Außerdem wird dadurch die Leistungsfähigkeit nur unwesentlich eingeschränkt.

Die Frage nach dem praktischen Nutzen des Frameworks bleibt aber noch immer bestehen. Schnell wurde bei der Umsetzung eines Projekts dieser Art klar, dass ein Trade-off zwischen einem ausreichenden Funktionsumfang und einer einfachen Handhabung getroffen werden muss. Auf Grund der hohen Anzahl unterschiedlicher, möglicher Spielgenres, die möglichst alle gleichermaßen von dem Framework profitieren sollten, war es erforderlich, alle Methoden offen oder lediglich als Interface zu implementieren. Dadurch entsteht während der Implementierung ein gewisser Mehraufwand bei der Anpassung. Können die Methoden allerdings direkt verwendet werden, wie es bei der Umsetzung kleinerer und einfacher Spiele, wie auch Pervasive Tic Tac Toe, häufig der Fall sein sollte, kann, gerade im Bereich der Nutzerverwaltung und bei der Realisierung des Regelwerks, viel Arbeitszeit eingespart werden.

Insgesamt kann man aber zu dem Schluß kommen, dass sich eine Spezialisierung auf einen einzelnen Spieltyp positiv für die Nutzbarkeit des Frameworks auswirken würde. Andererseits können auch bereits die rein konzeptuellen Anteile dieser Arbeit die Entwicklung von mobilen ortsbasierten Browserspielen vorantreiben, da Denkanstöße gegeben werden, welche Möglichkeiten in dieser noch kaum erforschten Spieleart stecken.

[1] siehe http://code.google.com/p/gwt2swf/wiki/GwtToSwfComunication

Kapitel 8

Zusammenfassung und Ausblick

Ergebnisse

In dieser Diplomarbeit wurde das mögliche Potenzial aufgezeigt, das in mobilen ortsbasierten Browserspielen, einer Kombination aus ortsbasierten Spielen und Browserspielen, steckt. Da es (bislang) keine Produkte dieser Art gibt, wurde daher auf die Besonderheiten beider Komponenten eingegangen, indem eine Vielzahl der bekanntesten Vertreter ausgiebig vorgestellt wurde. Da für diese Art der Software ein mobiles Gerät, das sich zum einen mit einem Server verbinden und zum anderen seine aktuelle Position bestimmen kann, benötigt wird, bedurfte es ebenfalls einer genauen Analyse, welche Techniken zur drahtlosen Kommunikation und zur Positionsbestimmung existieren. Des Weiteren mussten verbreitete Programmiersprachen im Umfeld der Webprogrammierung auf ihren möglichen Nutzen für diese Spiele hin untersucht werden. Schnell fiel auf, dass es, für eine hohe Kompatibilität zu möglichst vielen unterschiedlichen Gerätetypen, einer standardisierten Schnittstelle für den Zugriff auf deren Kontextinformationen bedarf. Diese wurde in DCCI gefunden, auch wenn es sich dabei zum Stand dieser Arbeit lediglich um eine Candidate Recommendation ohne weite Verbreitung handelt.

Nachdem damit eine ausreichende Wissensbasis über die Teilaspekte der mobilen ortsbasierten Browserspielen geschaffen wurde, konnten mögliche Gemeinsamkeiten unterschiedlichster Spieltypen herausgearbeitet werden, aus denen sich Anforderungen ableiten ließen. Diese Anforderungen bilden dabei einen Rahmen für ein möglichst offenes Framework, das die Entwicklung eines Spiels unterstützen kann, ohne dabei unnötig restriktiv zu sein und dennoch für alle Genres gleichermaßen nützlich ist.

Aufbauend auf diese Spezifikation wurde ein Framework modelliert und anschließend implementiert. Dabei kam in erster Linie das gwt mit einigen Erweiterungen - beispielsweise das Google Maps GWT API - zum Einsatz, da auf diese Weise sowohl der Code des Servers, als auch der des Clients nahezu vollständig in Java geschrieben werden konnte. Dennoch musste sich auch mit CSS und HTML beschäftigt werden, um das Erscheinungsbild der neu entwickelten Widgets gestalten zu können. Da die Analyse der führenden Browserspiele ergab, dass bei diesen häufig auch Flashanwendungen zum Einsatz kommen, musste dies beim Framework nicht nur berücksichtigt werden, sondern es wurde ebenfalls ein einfaches Quiz in Flash implementiert. Des Weiteren war für eine persistente Datenhaltung der Spielobjekte der Einsatz von Hibernate / Gilead nahezu unumgänglich.

Um die Funktionalität des Frameworks demonstrieren zu können, wurde mit Pervasive Tic Tac Toe ein Konzept eines mobilen ortsbasierten Browserspiels entwickelt und anschießend mit den

Komponenten des Frameworks umgesetzt. Zusätzlich wurden mögliche Variationen dieses Spiels vorgestellt, deren Implementierung allerdings kein Teil dieser Arbeit darstellt.

Abschließend wurden dem Framework die Anforderungen gegenübergestellt, um eine kritische Bewertung zu ermöglichen. Auch die Erfahrungen, die bei der Entwicklung des Spiels gewonnen wurden, flossen in diese Bewertung mit ein, um den praktischen Nutzen des Frameworks evaluieren zu können.

Ausblick

Obgleich die Beschreibung des Frameworks einen finalen Eindruck vermittelt, so ergeben sich auf diesem Gebiet, neben allgemeine Optimierungen am Quellcode, viele Möglichkeiten zur Erweiterung und Verbesserung, da der Schwerpunkt dieser Arbeit auf der Entwicklung eines Konzepts lag und die Implementierung lediglich zu dessen Demonstration beitragen sollte. Denkbare und wünschenswerte Verbesserungen wären:

- Erweiterung der Einstellmöglichkeiten durch die vorgestellte XML-Konfigurationsdatei
- Grafische Optimierung der erstellten Widgets
- Implementierung der Schnittstellen des Frameworks für unterschiedliche Datendanksysteme
- Implementierung einer GUI für einen Karteneditor
- Implementierung einer Oberfläche für die Konfiguration des Frameworks mit Hilfe der XML-Konfigurationsdatei
- Implementierung eines eigenen Chat-Services
- Implementierung eines eigenen Forums
- Implementierung eines Voice-Chats
- Optimierung der Hibernate Anfragen und Mappings
- Verbesserung des Login-Verfahrens (beispielsweise durch den Einsatz von OpenID[1])

Auch im Bereich des Prototyps wären neben allgemeinen Optimierungen an dem Quellcode und an der optischen Gestaltung, die Umsetzung der vorgestellten Varianten wünschenswert, da mit diesen noch mehr Aspekte der mobilen ortsbasierten Browserspielen vorgestellt werden können.

[1] siehe http://openid.net/

Anhang A

Benötigte Java-Bibliotheken

Um sicherzustellen, dass es dem Leser möglich ist, die im Rahmen dieser Arbeit entwickelten Software auf dem eigenen System kompilieren und ausführen zu können, werden in der Tabelle A.1 alle dafür verwendeten Bibliotheken, und in welchem Projekt - samt Versionsnummer - diese gefunden werden können, genannt. Außerdem kann ein fehlerfreier Einsatz des Frameworks nur in Kombination mit dem JDK 6 in der Version 1.6.0_07-b06 für eine x86 Architektur garantiert werden. Für die implementierte Hibernate-Schnittstelle wurde eine MySQL-Datenbank in der Version 5.1.30 für Windows x64 verwendet.

Projekt	**Version**	**Bibliothek**
ANTLR v2	2.7.6	`antlr-2.7.6.jar`
Apache log4j	1.2.15	`log4j-1.2.15.jar`
ASM	3.1	`asm-3.1.jar`
Beablib	3.3.0 Beta 21	`beanlib-3.3.0beta21b.jar` `beanlib-hibernate-3.3.0beta21b.jar`
cglib	2.2	`cglib-2.2.jar`
Commons-Collections	3.2.1	`commons-collections-3.2.1.jar`
dom4j	1.6.1	`dom4j-1.6.1.jar`

Gilead	1.2 RC1	`adapter4gwt-1.2.0.29.jar`
		`adapter-core-1.2.0.29.jar`
		`hibernate-util-1.2.0.29.jar`

Google Web Toolkit	1.5.3	`gwt-dev-windows.jar` `gwt-user.jar`
GWT - Google Maps API	2.0.0	`googlemaps_gwt_2_0_0.jar`
gwt-fabridge	1.2.1	`gwt-fabridge.jar` `gwt2swf-0.6.0.jar`
Java Transaction API	1.1	`jta-1.1.jar`
JavaMail API	1.4.1	`dsn.jar` `imap.jar`

		mailapi.jar pop3.jar smtp.jar
Javassist	3.4 GA	javassist-3.4.GA.jar
JBoss	5.0.0 GA	ejb3-persistence.jar
JDOM	1.1	jdom.jar
MySQL Connector/J	5.1.7	mysql-connector-java-5.1.7-bin.jar
SLF4J	1.5.6	slf4j-api-1.5.6.jar slf4j-log4j12-1.5.6.jar
Spring Framework	2.5.6	hibernate3.jar hibernate-annotations.jar hibernate-commons-annotations.jar hibernate-entitymanager.jar spring.jar spring-aop.jar
Xerces Java Parser	2.9.0	xercesImpl.jar

Tabelle A.1.: Zusammenstellung der eingesetzten Java-Bibliotheken

Anhang B

Datenbank-Schemata

Die vierzehn von Hibernate automatisch erstellten Tabellen zur Abbildung der in Abschnitt 5.3.1 eingeführten Klassen, deren Zusammenhänge in Abbildung 5.5 zu sehen sind, werden in den Tabellen B.1 - B.14 dargestellt. Neben dem Spaltennamen wird auch der SQL-Variablentyp und weitere Attribute, wie Primärschlüssel, Fremdschlüssel oder `not NULL`, angegeben.

Spaltenname	**Datentyp**	**Not NULL**
id (FK)	BIGINT(20)	true
radius	BIGINT(20)	true

Tabelle B.1.: `circles`-Tabelle

Spaltenname	**Datentyp**	**Not NULL**
id	BIGINT(20)	true
containersize	INTEGER	false
fillingdegree	INTEGER	false

Tabelle B.2.: `container`-Tabelle

Spaltenname	**Datentyp**	**Not NULL**
friendsetid (FK)	BIGINT(20)	true
friendid (FK)	BIGINT(20)	true

Tabelle B.3.: `friends`-Tabelle

Spaltenname	Datentyp	Not NULL
id (FK)	BIGINT(20)	true
name (UNIQUE)	VARCHAR(255)	true
iconurl	VARCHAR(255)	false
static	CHAR(1)	true
solid	CHAR(1)	true
latitude	BIGINT(20)	true
longitude	BIGINT(20)	true

Tabelle B.4.: `gobjects`-Tabelle

Spaltenname	Datentyp	Not NULL
id (FK)	BIGINT(20)	true
owner (FK)	BIGINT(20)	false
collectable	CHAR(1)	true
amount	INTEGER	true
itemid (FK)	BIGINT(20)	true

Tabelle B.5.: `items`-Tabelle

Spaltenname	Datentyp	Not NULL
membersetid (FK)	BIGINT(20)	true
memberid (FK)	BIGINT(20)	true

Tabelle B.6.: `members`-Tabelle

Spaltenname	Datentyp	Not NULL
objectid (FK)	BIGINT(20)	true
attributevalue	VARCHAR(255)	true
attributekey	VARCHAR(255)	true

Tabelle B.7.: `objectattributes`-Tabelle

Spaltenname	Datentyp	Not NULL
id	BIGINT(20)	true
name (UNIQUE)	VARCHAR(255)	true
mode	INTEGER	false
createdate	DATETIME	true
lastlogin	DATETIME	true
playtime	BIGINT(20)	false

Tabelle B.8.: `player`-Tabelle

Spaltenname	Datentyp	Not NULL
playerid (FK)	BIGINT(20)	true
attributevalue	VARCHAR(255)	true
attributekey	VARCHAR(255)	true

Tabelle B.9.: `playerattributes`-Tabelle

Spaltenname	Datentyp	Not NULL
id (FK)	BIGINT(20)	true
leftupperlatitude	BIGINT(20)	true
leftupperlongitude	BIGINT(20)	true
rightbottomlatitude	BIGINT(20)	true
rightbottomlongitude	BIGINT(20)	true

Tabelle B.10.: `rectangle`-Tabelle

Spaltenname	Datentyp	Not NULL
id (FK)	BIGINT(20)	true
account (FK)	BGINT(20)	false

Tabelle B.11.: `singleplayer`-Tabelle

Spaltenname	Datentyp	Not NULL
id (FK)	BIGINT(20)	true

Tabelle B.12.: `teams`-Tabelle

Spaltenname	Datentyp	Not NULL
id	BIGINT(20)	true
authorisations	VARCHAR(255)	true
name	VARCHAR(255)	true
password	VARCHAR(255)	true
firstname	VARCHAR(255)	true
lastname	VARCHAR(255)	true
email	VARCHAR(255)	false
createdate	DATETIME	true
lastlogin	DATETIME	true
playtime	BIGINT(20)	false
won	INTEGER	false
loss	INTEGER	false
draw	INTEGER	false

Tabelle B.13.: `user`-Tabelle

Spaltenname	Datentyp	Not NULL
userid (FK)	BIGINT(20)	true
attributevalue	VARCHAR(255)	true
attributekey	VARCHAR(255)	true

Tabelle B.14.: `userattributes`-Tabelle

Anhang C

Die XML-Konfigurationsdatei

Für die Konfiguration des Frameworks wurde ein XML-Schema erstellt, mit dessen Hilfe in der vorliegenden Version lediglich die Mail-Schnittstelle konfiguriert werden kann. Dieses Schema wird im Folgenden abgebildet. Zusätzlich wird in dem darauffolgenden Ausdruck, zum besseren Verständnis, eine dazu gültige XML-Datei angegeben.

Die XML-Schema-Definition

```
<!-- framework.cfg.xsd -->
<?xml version="1.0" encoding="ISO-8859-1"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="thesis">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="framework" minOccurs="1" maxOccurs="1">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="configuration"
                                                             minOccurs="1"
                                                                maxOccurs="1">
                                                                <xs:complexType>
                                                                        <xs:complexContent>
                                                                                <xs:extension
                                                                                     base="configuration_type"
                                                                                     />
                                                                        </xs:complexContent>
                                                                </xs:complexType>
                                                        </xs:element>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:complexType name="configuration_type">
                <xs:sequence>
                        <xs:element name="mail" minOccurs="1" maxOccurs="1">
                                <xs:complexType>
                                        <xs:sequence>
                                                <xs:element name="host" minOccurs="1" maxOccurs="1">
```

```
32                                                      <xs:complexType>
33                                                              <xs:attribute name="value"
                                                                    type="xs:string" />
34                                                      </xs:complexType>
35                                              </xs:element>
36                                              <xs:element name="login" minOccurs="1" maxOccurs="1">
37                                                      <xs:complexType>
38                                                              <xs:attribute name="value"
                                                                    type="xs:string" />
39                                                      </xs:complexType>
40                                              </xs:element>
41                                              <xs:element name="password" minOccurs="1"
                                                    maxOccurs="1">
42                                                      <xs:complexType>
43                                                              <xs:attribute name="value"
                                                                    type="xs:string" />
44                                                      </xs:complexType>
45                                              </xs:element>
46                                              <xs:element name="smtpauth" minOccurs="1"
                                                    maxOccurs="1">
47                                                      <xs:complexType>
48                                                              <xs:attribute name="value"
                                                                    type="xs:string" />
49                                                      </xs:complexType>
50                                              </xs:element>
51                                      </xs:sequence>
52                              </xs:complexType>
53                      </xs:element>
54              </xs:sequence>
55      </xs:complexType>
56  </xs:schema>
```

Eine gültige Konfigurationsdatei

```
1   <!-- framework.cfg.xml -->
2   <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
3
4   <thesis xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5           xsi:noNamespaceSchemaLocation="framework.cfg.xsd">
6           <framework>
7                   <configuration>
8                           <mail>
9                                   <host value="smtp_server.host.de" />
10                                  <login value="Username@host.de" />
11                                  <password value="password" />
12                                  <smtpauth value="true" />
13                          </mail>
14                  </configuration>
15          </framework>
16  </thesis>
```

Anhang C

Literaturverzeichnis

[AAH+97] Abowd, Gregory D. ; Atkeson, Christopher G. ; Hong, Jason ; Long, Sue ; Kooper, Rob ; Pinkerton, Mike: Cyberguide: a mobile context-aware tour guide. In: *ACM Wireless Networks* 3 (1997), Nr. 5, S. 421–433. http://dx.doi.org/http://dx.doi.org/10.1023/A:1019194325861. – DOI http://dx.doi.org/10.1023/A:1019194325861 (Zitiert auf Seite 28)

[Agg08] Agge: *Re: Informationen über Travian*. Email des Community Managers von Travian, September 2008 (Zitiert auf Seite 58)

[agl07] aglaforge: *GWT - Google Maps API*. Auf den Seiten von SourceForge.net. http://gwt.sourceforge.net/. Version: June 2007 (Zitiert auf Seite 72)

[AKY07] Acharya, Debopam ; Kumar, Vijay ; Yang, Gi-Chul: DAYS mobile: a location based data broadcast service for mobile users. In: *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*. New York, NY, USA : ACM, 2007, S. 901–905 (Zitiert auf Seite 65)

[AM00] Abowd, Gregory D. ; Mynatt, Elizabeth D.: Charting past, present, and future research in ubiquitous computing. In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 7 (2000), Nr. 1, S. 29–58. http://dx.doi.org/http://doi.acm.org/10.1145/344949.344988. – DOI http://doi.acm.org/10.1145/344949.344988 (Zitiert auf Seite 25)

[Aus03] Austin, Kenneth: Online gaming applications. In: *SIGGRAPH '03: ACM SIGGRAPH 2003 Web Graphics*. New York, NY, USA : ACM, 2003, S. 1–1 (Zitiert auf Seite 73)

[Azu95] Azuma, Ronald T.: *A survey of augmented reality*. citeseer.ist.psu.edu/article/azuma97survey.html. Version: 1995 (Zitiert auf Seite 26)

[Bat02] Bates, Bob: *Game design : Konzepte, Kreation, Vermarktung*. 1. Aufl. Düsseldorf : Sybex-Verlag, 2002. – 1–296 S. (Zitiert auf den Seiten 21 und 53)

[BBG+00] Banavar, Guruduth ; Beck, James ; Gluzberg, Eugene ; Munson, Jonathan ; Sussman, Jeremy ; Zukowski, Deborra: Challenges: an application model for pervasive computing. In: *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA : ACM, 2000, S. 266–274 (Zitiert auf den Seiten 17 und 25)

[BCF+06] Benford, Steve ; Crabtree, Andy ; Flintham, Martin ; Drozd, Adam ; Anastasi, Rob ; Paxton, Mark ; Tandavanitj, Nick ; Adams, Matt ; Row-Farr, Ju: Can you see me now? In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 13 (2006), Nr. 1, S.

100–133. http://dx.doi.org/http://doi.acm.org/10.1145/1143518.1143522. – DOI http://doi.acm.org/10.1145/1143518.1143522 (Zitiert auf Seite 40)

[BCN06a] Buttussi, Fabio ; Chittaro, Luca ; Nadalutti, Daniele: Bringing mobile guides and fitness activities together: a solution based on an embodied virtual trainer. In: *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. New York, NY, USA : ACM, 2006, S. 29–36 (Zitiert auf Seite 45)

[BCN06b] Buttussi, Fabio ; Chittaro, Luca ; Nadalutti, Daniele: *Mobile applications for health, fitness and sports*. http://hcilab.uniud.it/projects/2006-01.html. Version: January 2006 (Zitiert auf Seite 45)

[BFD+04] Benford, Steve ; Flintham, Martin ; Drozd, Adam ; Anastasi, Rob ; Rowland, Duncan ; Tandavanitj, Nick ; Adams, Matt ; Row-Farr, Ju ; Oldroyd, Amanda ; Sutton, Jon: Uncle Roy All Around You: Implicating the City in a Location-Based Performance. In: *Proc Advanced Computer Entertainment at ACE 2004*. Singapore : ACM Press, June 2004 (Zitiert auf Seite 51)

[BFHL01] Björk, Staffan ; Falk, Jennica ; Hansson, Rebecca ; Ljungstrand, Peter: Pirates! - Using the Physical World as a Game Board. In: *Interact 2001, IFIP TC.13 Conference on Human-Computer Interaction*, 2001 (Zitiert auf Seite 46)

[Bie05] Biere, Dr. J.: *REX Regensburg : Multimedia, Ausstellung, Film, Veranstaltungen, Museum : REXblog*. http://www.rex-regensburg.de/rexblog/. Version: 2005 (Zitiert auf Seite 47)

[Big08] Bigpoint: *Bigpoint presents Dancestar Online*. http://www.dancestar-online.de/. Version: September 2008 (Zitiert auf den Seiten 56 und 57)

[BKB+07] Ballagas, Rafael A. ; Kratz, Sven G. ; Borchers, Jan ; Yu, Eugen ; Walz, Steffen P. ; Fuhr, Claudia O. ; Hovestadt, Ludger ; Tann, Martin: REXplorer: a mobile, pervasive spell-casting game for tourists. In: *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2007, S. 1929–1934 (Zitiert auf Seite 47)

[BM08a] Blast Theory ; Mixed Reality Lab: *Can You See Me Now?* Auf den Seiten von Blast Theory. http://www.blasttheory.co.uk/bt/work_cysmn.html. Version: July 2008 (Zitiert auf Seite 41)

[BM08b] Blast Theory ; Mixed Reality Lab: *Uncle Roy All Around You*. Auf den Seiten von Blast Theory. http://www.blasttheory.co.uk/bt/work_uncleroy.html. Version: July 2008 (Zitiert auf Seite 50)

[BMB03] Blast Theory ; Mixed Reality Lab ; BT Exact: *Uncle Roy All Around You*. Auf den Seiten von Uncle Roy All Around You. http://www.uncleroyallaroundyou.co.uk/. Version: 2003 (Zitiert auf Seite 51)

[BML05] Benford, Steve ; Magerkurth, Carsten ; Ljungstrand, Peter: Bridging the physical and digital in pervasive gaming. In: *Communications of the ACM* 48 (2005), Nr. 3, S. 54–57. http://dx.doi.org/http://doi.acm.org/10.1145/1047671.1047704. – DOI http://doi.acm.org/10.1145/1047671.1047704 (Zitiert auf den Seiten 18 und 40)

[BNSM08] Brodt, Andreas ; Nicklas, Daniela ; Sathish, Sailesh ; Mitschang, Bernhard: Context-Aware Mashups for Mobile Devices. In: *Web Information Systems Engineering - WISE 2008 9th International Conference on Web Information Systems Engineering, Auckland, New Zealand, September 1-3, 2008, Proceedings*, Springer-Verlag, Januar 2008 (Lecture Notes in Computer Science) (Zitiert auf Seite 76)

[Bro04] Brownlow, Martin: *Goldene Regeln der Spieleprogrammierung : Methoden zur effizienten Programmierung von Computerspielen*. München; Wien : Hanser, 2004. – 1–270 S. (Zitiert auf Seite 23)

[Bro08] Brodt, Andreas: *Telar DCCI*. Auf den Seiten von maemo.org. http://telardcci.garage.maemo.org. Version: July 2008 (Zitiert auf Seite 76)

[BSF+04] Benford, Steve ; Seager, Will ; Flintham, Martin ; Anastasi, Rob ; Rowland, Duncan ; Humble, Jan ; Stanton, Danae ; Bowers, John ; Tandavanitj, Nick ; Adams, Matt ; Row-Farr, Ju ; Oldroyd, Amanda ; Sutton, Jon: The Error of Our Ways: The Experience of Self-Reported Position in a Location-Based Game. In: Davies, Nigel (Hrsg.) ; Mynatt, Elizabeth D. (Hrsg.) ; Siio, Itiro (Hrsg.): *Ubicomp* Bd. 3205, Springer, 2004 (Lecture Notes in Computer Science), 70–87 (Zitiert auf den Seiten 50 und 51)

[Bur07] Burns, John G.: *Pucman*. Auf den Seiten von EAGER. http://eager.back2roots.org/DATA/P/PUCMA.html. Version: November 2007 (Zitiert auf Seite 43)

[BZ98] Bokun, Igor ; Zielinski, Krzysztof: Active Badges — The Next Generation. In: *Linux Journal* 54 (1998), October, 24–26, 28–29. http://www.linuxjournal.com/article/3047 (Zitiert auf Seite 27)

[CDM+00] Cheverst, Keith ; Davies, Nigel ; Mitchell, Keith ; Friday, Adrian ; Efstratiou, Christos: Developing a context-aware electronic tourist guide: some issues and experiences. In: *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2000, S. 17–24 (Zitiert auf Seite 29)

[CFG+03] Cheok, Adrian D. ; Fong, Siew W. ; Goh, Kok H. ; Yang, Xubo ; Liu, Wei ; Farzbiz, Farzam: Human Pacman: a sensing-based mobile entertainment system with ubiquitous computing and tangible interaction. In: *NetGames '03: Proceedings of the 2nd workshop on Network and system support for games*. New York, NY, USA : ACM, 2003, S. 106–117 (Zitiert auf Seite 44)

[CGL+04] Cheok, Adrian D. ; Goh, Kok H. ; Liu, Wei ; Farbiz, Farzam ; Fong, Siew W. ; Teo, Sze L. ; Li, Yu ; Yang, Xubo: Human Pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. In: *Personal and Ubiquitous Computing* 8 (2004), Nr. 2, S. 71–81. http://dx.doi.org/http://dx.doi.org/10.1007/s00779-004-0267-x. – DOI http://dx.doi.org/10.1007/s00779–004–0267–x (Zitiert auf Seite 43)

[CL08] Cheok, Adrian D. ; Li, Yue: Ubiquitous interaction with positioning and navigation using a novel light sensor-based information transmission system. In: *Personal and Ubiquitous Computing* 12 (2008), Nr. 6, S. 445–458. http://dx.doi.org/http://dx.doi.org/10.1007/s00779-007-0140-9. – DOI http://dx.doi.org/10.1007/s00779–007–0140–9 (Zitiert auf Seite 66)

[CMD02] Cheverst, Keith ; Mitchell, Keith ; Davies, Nigel: The role of adaptive hypermedia in a context-aware tourist GUIDE. In: *Communications of the ACM* 45 (2002), Nr. 5, S. 47–51. http://dx.doi.org/http://doi.acm.org/10.1145/506218.506244. – DOI http://doi.acm.org/10.1145/506218.506244 (Zitiert auf Seite 29)

[Coh97] Cohen, Yosef ; Ryan, Tim (Hrsg.): *JavaScript cookbook*. New York, NY [u.a.] : John Wiley & Sons, inc., 1997. – 1–683 S. (Zitiert auf Seite 69)

[Com02] Computer Laboratory, University of Cambridge: *The Video Collection*. Auf den Seiten der AT&T Laboratories Cambridge. http://www.cl.cam.ac.uk/research/dtg/attarchive/labvid.html. Version: 2002 (Zitiert auf Seite 27)

[como8] COMUNIO.NET: *Fussball-Manager COMUNIO*. Auf den Seiten von comunio.de. http://www.comunio.de. Version: August 2008 (Zitiert auf den Seiten 55 und 56)

[Daio8] DAIMLER AG: *Cedy's World - Mercedes Benz for Kids*. Auf den Seiten der Daimler AG. http://www.cedysworld.com. Version: September 2008 (Zitiert auf Seite 54)

[Dan97] DANA, Peter H.: Global Positioning System (GPS) Time Dissemination for Real-Time Applications. In: *Real-Time Systems* 12 (1997), Nr. 1, S. 9–40. http://dx.doi.org/http://dx.doi.org/10.1023/A:1007906014916. – DOI http://dx.doi.org/10.1023/A:1007906014916 (Zitiert auf Seite 65)

[Deeo5] DEEG, Alex: *Ëpidemic Menace Pervasive Gaming-Event am Fraunhofer-Institut FIT*. Fraunhofer FIT, Informationsdienst Wissenschaft Pressemitteilung. http://idw-online.de/pages/de/news125867. Version: August 2005 (Zitiert auf den Seiten 41 und 42)

[Deeo6] DEEG, Alex: *Epidemic Menace – die zweite! CrossMedia-Spiel erfolgreich im Härte-Test*. Auf den Seiten des Fraunhofer-Institut für Angewandte Informationstechnik. http://www.fit.fraunhofer.de/lenya/fit2006/live/presse/presse2006/06-07-14_de.html. Version: July 2006 (Zitiert auf Seite 42)

[Disoo] DISTRIBUTED MULTIMEDIA RESEARCH GROUP: *Guide News Updates*. Auf den Seiten des Computing Department, Lancaster University. http://www.guide.lancs.ac.uk/screenshots.html. Version: August 2000 (Zitiert auf Seite 29)

[Doo5] DO, Norman: How to Win at TicTacToe. In: *The Australian Mathematical Society* 32 (2005), July, Nr. 3, 151 – 161. http://www.austms.org.au/Gazette/2005/Jul05/mathellaneous.pdf (Zitiert auf Seite 108)

[EEL+05] EKMAN, Inger ; ERMI, Laura ; LAHTI, Jussi ; NUMMELA, Jani ; LANKOSKI, Petri ; MÄYRÄ, Frans: Designing sound for a pervasive mobile game. In: *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. New York, NY, USA : ACM, 2005, S. 110–116 (Zitiert auf den Seiten 48 und 49)

[EKo8] ERNESTI, Johannes ; KAISER, Peter: *Python : das umfassende Handbuch*. 1. Auflage. Bonn : Galileo Press, 2008 (Galileo Computing). – 1–819 S. http://galileocomputing.de/openbook/python/index.htm (Zitiert auf Seite 69)

[EMo5] ERMI, Laura ; MÄYRÄ, Frans: Challenges for pervasive mobile game design: examining players' emotional responses. In: *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. New York, NY, USA : ACM, 2005, S. 371–372 (Zitiert auf Seite 49)

[Equo8] EQUATOR: *Uncle Roy All Around You*. Auf den Seiten von Equator. http://www.equator.ac.uk/index.php/articles/619. Version: 2008 (Zitiert auf Seite 50)

[FAB+03] FLINTHAM, Martin ; ANASTASI, Rob ; BENFORD, Steve ; DROZD, Adam ; MATHRICK, James ; ROWLAND, Duncan ; OLDROYD, Amanda ; SUTTON, Jon ; TANDAVANITJ, Nick ; ADAMS, Matt ; ROW-FARR, Ju: Uncle Roy all around you: mixing games and theatre on the city streets. In: *DIGRA Conf.*, 2003, 168–177 (Zitiert auf Seite 50)

[fabo6] FABIAN: *Comunio - Klicken vorm Kicken*. Auf den Seiten von Fudder.de. http://fudder.de/artikel/2006/04/20/klicken-vorm-kicken/. Version: April 2006 (Zitiert auf Seite 55)

[FBA+03] FLINTHAM, Martin ; BENFORD, Steve ; ANASTASI, Rob ; HEMMINGS, Terry ; CRABTREE, Andy ; GREENHALGH, Chris ; TANDAVANITJ, Nick ; ADAMS, Matt ; ROW-FARR, Ju: Where on-line meets on the streets: experiences with mobile mixed reality games. In: *CHI '03:*

Proceedings of the SIGCHI conference on Human factors in computing systems. New York, NY, USA : ACM, 2003, S. 569–576 (Zitiert auf Seite 41)

[FLBH01] Falk, Jennica ; Ljungstrand, Peter ; Björk, Staffan ; Hansson, Rebecca: Pirates: proximity-triggered interaction in a multi-player game. In: *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2001, S. 119–120 (Zitiert auf Seite 47)

[FLS06] Fischer, Joel ; Lindt, Irma ; Stenros, Jaakko ; Fraunhofer Institut für Angewandte Informationstechnik - IPerG (Hrsg.): *Final Crossmedia report (part II) – Epidemic Menace II Evaluation Report*. 0.1. Fraunhofer-Institut für Angewandte Informationstechnik FIT Schloss Birlinghoven 53754 Sankt Augustin, Germany: Fraunhofer Institut für Angewandte Informationstechnik - IPerG, December 2006. http://www.pervasive-gaming.org/Deliverables/D8.8-Part-II.pdf (Zitiert auf Seite 42)

[Fri05] Friedburg, Sven O.: *Möglichkeiten ortsabhängiger Interaktionen auf Basis mobiler Endgeräte*, HAW Hamburg, Studienarbeit, May 2005. http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/studien/friedburg.pdf (Zitiert auf Seite 65)

[Fri08] Frickel, Claudia: *Wii Fit: Die Fitness-Konsole im Test*. Auf den Seiten von FOKUS Online. http://www.focus.de/digital/games/spielkonsolen/tid-9718/wii-fit-die-fitness-konsole-im-test_aid_297125.html. Version: April 2008 (Zitiert auf Seite 44)

[Gam08] GamesSphere Redaktion: *Dancestar Online Preview @ GamesSphere*. Auf den Seiten von GamesSphere. http://www.gamessphere.de/index.html?article=Dancestar_Online_Preview. Version: August 2008 (Zitiert auf Seite 56)

[Gol05] Golem.de: *Gegen Online-(Rollen-)Spielsucht: China beschränkt Spielzeit*. Aus dem Nachrichtenarchiv von Golem.de. http://www.golem.de/0508/40054.html. Version: August 2005 (Zitiert auf Seite 84)

[Hö07] Hüsing, Alexander: *Vorreiter Deutschland Browserspiele erobern das Web*. Auf den Seiten von n-tv.de. http://www.n-tv.de/778191.html. Version: March 2007 (Zitiert auf Seite 52)

[Har07] Harmonix Music Systems: *Rock Band*. http://www.rockband.com/. Version: 2007 (Zitiert auf Seite 17)

[HHW+04] Hors, Arnaud L. ; Hégaret, Philippe L. ; Wood, Lauren ; Nicol, Gavin ; Robie, Jonathan ; Champion, Mike ; Byrne, Steve: *Document Object Model (DOM) Level 3 Core Specification*. W3C Recommendation. http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/. Version: April 2004 (Zitiert auf Seite 75)

[Hil07] Hilgert, Chris: *YETISPORTS, the official Site of the new Online-Game*. Auf den Seiten der ROOT9 MediaLab GmbH. http://www.yetisports.org/. Version: December 2007 (Zitiert auf den Seiten 60 und 61)

[HKL+99] Hohl, Fritz ; Kubach, Uwe ; Leonhardi, Alexander ; Rothermel, Kurt ; Schwehm, Markus: Next century challenges: Nexus—an open global infrastructure for spatial-aware applications. In: *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA : ACM, 1999, S. 249–255 (Zitiert auf Seite 34)

[HPS06] Heinle, Nick ; Peña, Bill ; Speidel, Ulrich: *Webdesign mit JavaScript & Ajax*. 2. Aufl. Beijing; Köln [u.a.] : O'Reilly Verlag GmbH & Co. KG, 2006 (O'Reillys basics). – 1–254 S. (Zitiert auf Seite 66)

[HS08] Hunter, William ; Slabihoud, Stephan: *The Dot Eaters*. Auf den Seiten des 8bit-Museums. http://www.8bit-museum.de/docs/play2sta4.htm. Version: January 2008 (Zitiert auf Seite 42)

[HT07] Hanson, Robert ; Tacy, Adam: *GWT im Einsatz - AJAX-Anwendungen entwickeln mit dem Google Web Toolkit*. München; Wien : Carl Hanser Verlag, 2007. – 1–543 S. (Zitiert auf den Seiten 66 und 74)

[Hui56] Huizinga, Johan: *Rowohlts deutsche Enzyklopädie*. Bd. 21: *Homo ludens : vom Ursprung der Kultur im Spiel*. Hamburg : Rowohlt, 1956. – 7–220 S. (Zitiert auf Seite 16)

[HW03] Hansen, Heiko ; Winterscheidt, Rolf: *Kabellose Netzwerke, Wireless LAN von Anfang an*. 1. Aufl. Köln : Sybex-Verlag, 2003. – 5–302 S. (Zitiert auf den Seiten 63, 64 und 65)

[HWR04] Hégaret, Philippe L. ; Wood, Lauren ; Robie, Jonathan: *What is the Document Object Model?* Auf den Seiten des W3C. http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/introduction.html. Version: April 2004 (Zitiert auf Seite 68)

[HWW05] Hégaret, Philippe L. ; Whitmer, Ray ; Wood, Lauren: *Document Object Model (DOM)*. W3C Recommendation. http://www.w3.org/DOM/. Version: January 2005 (Zitiert auf Seite 68)

[Hyp04] Hyper Madialab ; MOGAME project (Hrsg.): *Songs of North - Pohjoisen Laulut*. Version 1. MOGAME project Hypermedia Laboratory 33014 University of Tampere, Finland: MOGAME project, October 2004. http://www.uta.fi/hyper/projektit/mogame/laulut_esite.pdf (Zitiert auf Seite 48)

[It'02] It's Alive: *BotFighters*. Aus dem Archiv von ARS Electronica. http://www.aec.at/de/archives/picture_ausgabe_03_new.asp?iAreaID=78&showAreaID=242&iImageID=33095. Version: 2002 (Zitiert auf Seite 39)

[Jö8] Jäger, Kai: *Ajax in der Praxis : Grundlagen, Konzepte, Lösungen*. Berlin; Heidelberg : Springer-Verlag, 2008. – 1–355 S. http://dx.doi.org/http://dx.doi.org/10.1007/978-3-540-69334-5. http://dx.doi.org/http://dx.doi.org/10.1007/978-3-540-69334-5 (Zitiert auf Seite 74)

[Kaa03] Kaasinen, Eija: User needs for location-aware mobile services. In: *Personal and Ubiquitous Computing* 7 (2003), Nr. 1, S. 70–79. http://dx.doi.org/http://dx.doi.org/10.1007/s00779-002-0214-7. – DOI http://dx.doi.org/10.1007/s00779–002–0214–7 (Zitiert auf Seite 27)

[KH06] Kehle, Markus ; Hien, Robert: *Hibernate und das Java Persistence API: Einstieg und professioneller Einsatz*. Frankfurt (Main) : Entwickler.press, 2006. – 5 – 269 S. (Zitiert auf Seite 72)

[KLRS99] Kubach, Dipl.-Inf. U. ; Leonhardi, Dipl.-Inf. A. ; Rothermel, Prof. Dr. K. ; Schwehm, Dr. M.: Analysis of Distribution Schemes for the Management of Location Information / Institut für Parallele und Verteilte Systeme, Abteilung Verteilte Systeme. Version: June 1999. http://elib.uni-stuttgart.de/opus/volltexte/1999/425/. Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR) Fakultät Informatik Universität Stuttgart Breitwiesenstr. 20 - 22 D-70565 Stuttgart : Universität Stuttgart, Fakultät Informatik, June 1999 (1). – Fakultätsbericht 1999 (Zitiert auf Seite 34)

[Lae01] Laerhoven, Kristof V.: *Technology for Enabling Awareness*. Auf den Seiten des Telecooperation Offices (TecO). http://www.teco.edu/tea/tea_hrd2.html. Version: April 2001 (Zitiert auf Seite 31)

[LHB06] Lampe, Matthias ; Hinske, Steve ; Brockmann, Sandra: Mobile Device based Interaction Patterns in Augmented Toy Environments. In: Strang, Thomas (Hrsg.) ; Cahill, Vinny (Hrsg.) ; Quigley, Aaron (Hrsg.): *Pervasive 2006 Workshop Proceedings (Third International Workshop on Pervasive Gaming Applications, PerGames 2006)*. Dublin, Ireland, May 2006, 109–118 (Zitiert auf Seite 38)

[LHN+04] Lankoski, Petri ; Heliö, Satu ; Nummela, Jani ; Lahti, Jussi ; Mäyrä, Frans ; Ermi, Laura: A case study in pervasive game design: the songs of north. In: *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*. New York, NY, USA : ACM, 2004, S. 413–416 (Zitiert auf Seite 49)

[LOPBG07] Lindt, Irma ; Ohlenburg, Jan ; Pankoke-Babatz, Uta ; Ghellal, Sabiha: A report on the crossmedia game epidemic menace. In: *Computers in Entertainment (CIE)* 5 (2007), Nr. 1, S. 1–8. http://dx.doi.org/http://doi.acm.org/10.1145/1236224.1236237. – DOI http://doi.acm.org/10.1145/1236224.1236237 (Zitiert auf Seite 42)

[Los08] Loschek, Fabian: *AW: Informationen über COMUNIO*. Email des Entwicklers von COMUNIO, Septermber 2008 (Zitiert auf Seite 56)

[Lub08] Lubek, Sven: *Browserspiele / Browsergames - Infos, Tests und Links zu den besten Browsergames. Kostenlos.* Auf den Seiten von Browserspiele1.de. http://www.browserspiele1.de. Version: September 2008 (Zitiert auf den Seiten 24 und 52)

[Mar08a] Marchesson, Bruno: *hibernate4gwt*. Auf den Seiten von http://hibernate4gwt.sourceforge.net. http://hibernate4gwt.sourceforge.net. Version: November 2008 (Zitiert auf Seite 73)

[Mar08b] Marchesson, Bruno: *Home - Gilead*. Auf den Seiten von http://noon.gilead.free.fr. http://noon.gilead.free.fr/gilead/. Version: November 2008 (Zitiert auf Seite 73)

[Mat01] Mattern, Friedemann: Pervasive / Ubiquitous Computing. In: *Informatik-Spektrum* 24 (2001), June, Nr. 3, 145–147. http://www.vs.inf.ethz.ch/publ/papers/UbiPvCSchlagwort.pdf (Zitiert auf den Seiten 15 und 25)

[MEM04] Magerkurth, Carsten ; Engelke, Timo ; Memisoglu, Maral: Augmenting the virtual domain with physical and social elements: towards a paradigm shift in computer entertainment technology. In: *Computers in Entertainment (CIE)* 2 (2004), Nr. 4, S. 1–20. http://dx.doi.org/http://doi.acm.org/10.1145/1037851.1037870. – DOI http://doi.acm.org/10.1145/1037851.1037870 (Zitiert auf Seite 35)

[Mey07] Meyers Lexikon online: *Spiel (Sachbegriffe)*. http://lexikon.meyers.de/meyers/Spiel_(Sachbegriffe). Version: 2007 (Zitiert auf Seite 16)

[Mic04] Michael: *Cedysworld*. Auf den Seiten der Medienpädagogik an der Universität Ulm. http://medienmami.uni-ulm.de/2004/06/25/cedysworld/. Version: June 2004 (Zitiert auf Seite 53)

[Mil73] Millar, Susanna: *Psychologie des Spiels*. 1. Aufl. Ravensburg : Otto Maier Verlag, 1973 (EGS-Texte). – 5–271 S. (Zitiert auf Seite 17)

[Mix08] Mixed Reality Lab: *Human Pacman*. http://www.mixedrealitylab.org/index.php?option=com_content&task=view&id=42&Itemid=74. Version: March 2008 (Zitiert auf den Seiten 17 und 43)

[ML07] Mintert, Stefan ; Leisegang, Christoph: *Ajax: Grundlagen, Frameworks und Praxislösungen*. 1. Auflage. Heidelberg : dpunkt.verlag GmbH, 2007 (IX-Edition). – 1–291 S. (Zitiert auf Seite 67)

[Muto7] Mutz, Uwe: *Flash CS3, AJAX & PHP*. München; Boston; San Francisco [u.a.] : Addison Wesley Verlag, 2007 (dpi). – 1–431 S. (Zitiert auf Seite 74)

[MZo8] Moussaoui, Hassan E. ; Zeppenfeld, Klaus ; Günther, Prof. Dr. O. (Hrsg.) ; Karl, Prof. Dr. W. (Hrsg.) ; Lienhart, Prof. Dr. R. (Hrsg.) ; Zeppenfeld, Prof. Dr. K. (Hrsg.): *AJAX : Geschichte, Technologie, Zukunft*. Berlin; Heidelberg : Springer-Verlag, 2008 (Informatik im Fokus). – 1–173 S. http://dx.doi.org/http://dx.doi.org/10.1007/978-3-540-73115-3. http://dx.doi.org/http://dx.doi.org/10.1007/978-3-540-73115-3 (Zitiert auf Seite 67)

[Napo6] Napitupulu, Jan: *Pervasive Gaming: Entwicklung von ortsabhängiger Spielesoftware*. 1. Aufl. Saarbrücken : VDM Verlag Dr. Müller, 2006. – 1–116 S. (Zitiert auf den Seiten 21, 23, 39, 40, 63 und 64)

[Navo8a] Navicore Ltd.: *Press Resources*. Auf den Seiten von Navicore Ltd. http://www.navicoretech.com/Corporate/Press/PressKit/en_GB/PressKit/. Version: July 2008 (Zitiert auf Seite 30)

[Navo8b] Navicore Ltd.: *Wayfinder brings free maps and award-winning mobile navigation to the new Nokia N810 with integrated GPS*. Auf den Seiten von Navicore Ltd. http://www.navicoretech.com/Consumer/News/news/de_DE/n810_announce/. Version: July 2008 (Zitiert auf Seite 30)

[Nefo6] Nefzger, Wolfgang: *CSS Cascading Style Sheets für Profis*. Poing : Franzis Verlag GmbH, 2006 (Franzis Professional Series). – 15–352 S. (Zitiert auf Seite 68)

[NGS+01] Nicklas, Daniela ; Grossmann, Matthias ; Schwarz, Thomas ; Volz, Steffen (ifp) ; Mitschang, Bernhard: A Model-Based, Open Architecture for Mobile, Spatially Aware Applications. In: Jensen, Christian S. (Hrsg.) ; Schneider, Markus (Hrsg.) ; Seeger, Bernhard (Hrsg.) ; Tsotras, Vassilis J. (Hrsg.): *Proceedings of the 7th International Symposium on Spatial and Temporal Databases: SSTD 2001; Redondo Beach, CA, USA, July 12-15, 2001* Bd. 2121. Berlin, Heidelberg, New York : Springer-Verlag, July 2001 (Lecture Notes in Computer Science), 117–135 (Zitiert auf Seite 32)

[Nino8] Nintendo: *Wii Fit*. http://www.nintendo.com/wiifit/launch/. Version: 2008 (Zitiert auf den Seiten 17 und 44)

[Noko8] Nokia: *Internet Tablet OS 2008 Edition - Funktionserweiterung Bedienungsanleitung*. 2 DE. Nokia Corporation P.O. Box 226, FI-00045 Nokia Group Finland, June 2008. http://nds1.nokia.com/phones/files/guides/Nokia_N810_OS2008_upgrade_UG_de.pdf (Zitiert auf Seite 18)

[NPMo1] Nicklas, Daniela ; Pfisterer, Christoph ; Mitschang, Bernhard: Towards Location-based Games. In: Sing, Alfred Loo W. (Hrsg.) ; Man, Wan H. (Hrsg.) ; Wai, Wong (Hrsg.) ; Ning, Cyril T. (Hrsg.): *Proceedings of the International Conference on Applications and Development of Computer Games in the 21st Century: ADCOG 21; Hongkong Special Administrative Region, China, November 22-23 2001*. Hong Kong : Division of Computer Studies, City University of Hong kong, Hong Kong SAR, China, November 2001, 61–67 (Zitiert auf Seite 35)

[OSYT98] Ohshima, Toshikazu ; Satoh, Kiyohide ; Yamamoto, Hiroyuki ; Tamura, Hideyuki: AR2 Hockey: A Case Study of Collaborative Augmented Reality. In: *VRAIS '98: Proceedings of the Virtual Reality Annual International Symposium*. Washington, DC, USA : IEEE Computer Society, 1998, S. 268–275 (Zitiert auf Seite 36)

[PCB00] Priyantha, Nissanka B. ; Chakraborty, Anit ; Balakrishnan, Hari: The Cricket location-support system. In: *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA : ACM, 2000, S. 32–43 (Zitiert auf Seite 33)

[Pro07] Produktdesign.com: *Cyberspacebrille - 3D Brille - Head Mounted Display*. http://www.produktdesign.com/cyberspacebrille.htm. Version: November 2007 (Zitiert auf Seite 26)

[PSY97] Perry, Lynellen D. S. ; Smith, Christopher M. ; Yang, Steven: An investigation of current virtual reality interfaces. In: *Crossroads* 3 (1997), Nr. 3, S. 23–28. http://dx.doi.org/http://doi.acm.org/10.1145/270974.270983. – DOI http://doi.acm.org/10.1145/270974.270983 (Zitiert auf Seite 26)

[PT02] Piekarski, Wayne ; Thomas, Bruce: ARQuake: the outdoor augmented reality gaming system. In: *Communications of the ACM* 45 (2002), Nr. 1, S. 36–38. http://dx.doi.org/http://doi.acm.org/10.1145/502269.502291. – DOI http://doi.acm.org/10.1145/502269.502291 (Zitiert auf Seite 36)

[Ram00] Ramirez, Ariel O.: Three-Tier Architecture. In: *Linux Journal* 75 (2000), July, Nr. 7, 7. http://www.linuxjournal.com/article/3508 (Zitiert auf Seite 91)

[Red01] Redaktion SELFHTML: *SELFHTML*. http://de.selfhtml.org/. Version: 8.1.2, October 2001 (Zitiert auf den Seiten 70 und 74)

[Rog96] Rogers, Steve: *Willkommen bei Esprit, dem Informationstechnologie-Programm*. Auf den Seiten von Community Research & Development Information Service (EUROPA - CORDIS). http://cordis.europa.eu/esprit/src/intro-d.htm. Version: December 1996 (Zitiert auf Seite 31)

[Ros08] RoseIndia.Net: *EJB Tutorial*. http://www.roseindia.net/ejb/. Version: 2008 (Zitiert auf Seite 70)

[Rot05] Roth, Jörg: *Mobile Computing: Grundlagen, Technik, Konzepte*. 2., aktualisierte Aufl. Heidelberg : dpunkt.verlag, 2005 (Zitiert auf Seite 65)

[Rus07] Russell, Ian: *Now, I can see you.* Auf den Seiten von iArchitectures (Zitiert auf Seite 41)

[SAG+93] Schilit, Bill N. ; Adams, Norman ; Gold, Rich ; Tso, Michael M. ; Want, Roy: The PARCTAB Mobile Computing System. In: *Workshop on Workstation Operating Systems*, 1993, 34–39 (Zitiert auf Seite 28)

[Sau08] Sauter, Martin: *Grundkurs Mobile Kommunikationssysteme*. 3., erweiterte Auflage. Wiesbaden : Friedr. Vieweg & Sohn Verlag | GWV Fachverlage GmbH, 2008. – 1–424 S. http://dx.doi.org/http://dx.doi.org/10.1007/978-3-8348-9445-8. http://dx.doi.org/http://dx.doi.org/10.1007/978-3-8348-9445-8 (Zitiert auf den Seiten 64 und 65)

[SBG99] Schmidt, Albrecht ; Beigl, Michael ; Gellersen, Hans-W.: There is more to context than location. In: *Computers and Graphics* 23 (1999), Nr. 6, 893–901. citeseer.ist.psu.edu/schmidt98there.html (Zitiert auf Seite 31)

[Sch02] Schmidt, Albrecht: *Ubiquitous Computing - Computing in Context*. Computing Department, Lancaster University, England, U.K., Lancaster University, A thesis submitted for the degree of Ph.D. in Computer Science, November 2002. http://www.comp.lancs.ac.uk/~albrecht/phd/Albrecht_Schmidt_PhD-Thesis_Ubiquitous-Computing_print1.pdf (Zitiert auf Seite 33)

[Sch08] Schöning, Uwe: *Theoretische Informatik - kurz gefasst*. 5. Auflage. Heidelberg : Spektrum Akademischer Verlag, 2008 (Hochschultaschenbuch) (Zitiert auf Seite 100)

[See08] Seemann, Michael: *Das Google Web Toolkit: GWT*. Dt. Orginalausgabe, 1. Auflage. Beijing, Köln [u.a.] : O'Reilly, 2008. – 1–253 S. (Zitiert auf Seite 72)

[SF99] Schmidt, Albrecht ; Forbess, Jessica: What GPS doesn't tell you: determining one's context with low-level sensors. In: *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on* 2 (1999), Sep, 883-886 vol.2. http://dx.doi.org/10.1109/ICECS.1999.813250. – DOI 10.1109/ICECS.1999.813250 (Zitiert auf Seite 30)

[SF06] Steyer, Ralph ; Fuchs, Dr. J.: *AJAX mit ASP.NET und Atlas*. München; Boston; San Francisco [u.a.] : Addison Wesley Verlag, 2006 (PROGRAMMER'S CHOICE). – 11–392 S. (Zitiert auf Seite 67)

[SHA+06] Schmidt, Albrecht ; Häkkilä, Jonna ; Atterer, Richard ; Rukzio, Enrico ; Holleis, Paul: Utilizing mobile phones as ambient information displays. In: *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 2006, S. 1295–1300 (Zitiert auf Seite 18)

[Sot02] Sotamaa, Olli: All The World's A Botfighter Stage: Notes on Location-based Multi-User Gaming. In: *Proceedings of the Computer Games and Digital Cultures Conference, June 6-8, 2002, Tampere, Finland*, 2002, 35–44 (Zitiert auf den Seiten 39 und 40)

[SOYT99] Satoh, Kiyohide ; Ohshima, Toshikazu ; Yamamoto, Hiroyuki ; Tamura, Hideyuki: Case studies of see-through augmentation in mixed reality project. In: *IWAR '98: Proceedings of the international workshop on Augmented reality : placing artificial objects in real scenes*. Natick, MA, USA : A. K. Peters, Ltd., 1999, S. 3–18 (Zitiert auf den Seiten 35 und 36)

[SRH05] Steinicke, Frank ; Ropinski, Timo ; Hinrichs, Klaus: A generic virtual reality software system's architecture and application. In: *ICAT '05: Proceedings of the 2005 international conference on Augmented tele-existence*. New York, NY, USA : ACM, 2005, S. 220–227 (Zitiert auf Seite 26)

[Sta08] Starforge Studios: *Dancestar Online - Browsergame*. Auf den Seiten von GamesSphere. http://www.gamessphere.de/games.php/browsergames/dancestar-online/744/. Version: August 2008 (Zitiert auf Seite 56)

[Ste06] Steyer, Ralph: *AJAX mit Java-Servlets und JSP*. München, Boston [u.a.] : Addison Wesley Verlag, 2006 (open source library). – 17–351 S. (Zitiert auf Seite 70)

[Ste07] Steyer, Ralph: *Google Web Toolkit : Ajax-Applikationen mit Java*. Unterhaching : entwickler press, 2007. – 13–320 S. (Zitiert auf Seite 71)

[SW95] Schilit, Bill ; Want, Roy: *PARCTAB Picture Gallery*. Auf den Seiten des Palo Alto Research Center. http://sandbox.xerox.com/parctab/pics.html. Version: April 1995 (Zitiert auf Seite 28)

[Sym07] Symode09: *A typical game of tic tac toe*. Auf den Seiten von Wikimedia. http://commons.wikimedia.org/wiki/File:Tic_tac_toe-js.svg. Version: May 2007 (Zitiert auf Seite 107)

[Tö05] Tötsches, Edith: *Adobe übernimmt Macromedia für 3,4 Milliarden Dollar*. Auf den Seiten des heise Verlags. http://www.heise.de/newsticker/Adobe-uebernimmt-Macromedia-fuer-3-4-Milliarden-Dollar--/meldung/58672. Version: April 2005 (Zitiert auf Seite 74)

[TCD+02] Thomas, Bruce ; Close, Ben ; Donoghue, John ; Squires, John ; Bondi, Phillip D. ; Piekarski, Wayne: First Person Indoor/Outdoor Augmented Reality Application: ARQuake. In: *Personal and Ubiquitous Computing* 6 (2002), Nr. 1, S. 75–86. http://dx.doi.org/http://dx.doi.org/10.1007/s007790200007. – DOI http://dx.doi.org/10.1007/s007790200007 (Zitiert auf Seite 37)

[Theo9] The Professional Open Source Company: *hibernate.org - Hibernate*. Auf den Seiten von hibernate.org. http://www.hibernate.org/. Version: January 2009 (Zitiert auf Seite 72)

[Tho98] Thorp, Edward O.: The Invention of the First Wearable Computer. In: *ISWC '98: Proceedings of the 2nd IEEE International Symposium on Wearable Computers*. Washington, DC, USA : IEEE Computer Society, 1998, 4–8 (Zitiert auf Seite 27)

[Trao8a] Travian Games GmbH: *Travian - Browserspiel - Römer, Gallier & Germanen*. http://www.travian.de/index.php. Version: September 2008 (Zitiert auf den Seiten 57 und 58)

[Trao8b] Travian Games GmbH: *Travianer*. http://www.travianer.de/. Version: September 2008 (Zitiert auf den Seiten 59 und 60)

[Tsuoo] Tsui, James Bao-Yen ; Chang, Kai (Hrsg.): *Fundamentals of global positioning system receivers : a software approach*. New York, NY [u.a.] : John Wiley & Sons, inc., 2000 (Wiley series in microwave and optical engineering). – 1–238 S. (Zitiert auf Seite 65)

[Unio7a] Universität Stuttgart: *Infosheets: ISIS - A City Guide Based on the Nexus Platform*. Auf den Seiten des Nexus Projekts der Universität Stuttgart. http://www.nexus.uni-stuttgart.de/de/forschung/dokumente/docs-context-workshop/infosheets-ISIS.pdf. Version: 2007 (Zitiert auf Seite 32)

[Unio7b] Universität Stuttgart: *Poster: Nexus Scout - An Advanced Location-Based Application*. Auf den Seiten des Nexus Projekts der Universität Stuttgart. http://www.nexus.uni-stuttgart.de/de/forschung/dokumente/docs-context-workshop/poster-NexusScout.pdf. Version: 2007 (Zitiert auf Seite 32)

[Volo2] Vollenweider, Martin: *Macromedia Flash MX und ActionScript*. Kilchberg : SmartBooks Publishing AG, 2002. – 20–296 S. (Zitiert auf Seite 74)

[Waloo] Waldron, Rick: *The Flash history*. Auf den Seiten des Flash Magazine. http://www.flashmagazine.com/news/detail/the_flash_history/. Version: November 2000 (Zitiert auf Seite 74)

[WBB+06] Walz, Steffen P. ; Ballagas, Rafael ; Borchers, Jan ; Mendoza, Joel ; Kratz, Sven ; Wartmann, Christoph ; Fuhr, Claudia ; Tann, Martin J. ; Shin, Dong Y. ; Hameed, Bilal ; Bardos, Laszlo ; Hovestadt, Ludger: Cell Spell-Casting: Designing a Locative and Gesture Recognition Multiplayer Smartphone Game for Tourists. In: *Proc. PERGAMES, Third International Workshop on Pervasive Gaming Applications at PERVASIVE 2006*. Dublin, Ireland : LNCS, May 2006 (Zitiert auf Seite 48)

[Weao8] Wearable Computer Lab - UniSA: *Projects - WCL - University of South Australia*. Auf den Seiten der University of South Australia. http://wearables.unisa.edu.au/Projects. Version: May 2008 (Zitiert auf Seite 37)

[Wei93] Weiser, Mark: Some computer science issues in ubiquitous computing. In: *Communications of the ACM* 36 (1993), Nr. 7, S. 75–84. http://dx.doi.org/http://doi.acm.org/10.1145/159544.159617. – DOI http://doi.acm.org/10.1145/159544.159617 (Zitiert auf Seite 25)

[Weno7] Wenz, Christian: *JavaScript und AJAX*. 7., aktualisierte Auflage. Bonn : Galileo Press, 2007 (Galileo Computing). – 1–839 S. http://www.galileocomputing.de/openbook/javascript_ajax/index.htm (Zitiert auf den Seiten 69 und 74)

[WHFG92] Want, Roy ; Hopper, Andy ; Falcão, Veronica ; Gibbons, Jonathan: The active badge location system. In: *ACM Transactions on Information Systems (TOIS)* 10 (1992), Nr. 1, S. 91–102. http://dx.doi.org/http://doi.acm.org/10.1145/128756.128759. – DOI http://doi.acm.org/10.1145/128756.128759 (Zitiert auf Seite 27)

[WHR+07] Waters, Keith ; Hosn, Rafah A. ; Raggett, Dave ; Sathish, Sailesh ; Womer, Matt ; Froumentin, Max ; Lewis, Rhys ; Rosenblatt, Keith: *Delivery Context: Client Interfaces (DCCI) 1.0*. W3C Candidate Recommendation. http://www.w3.org/TR/DPF/. Version: December 2007 (Zitiert auf Seite 76)

[Wi-07] Wi-Fi Alliance: *Knowledge Center - WPA2™ (Wi-Fi Protected Access 2)*. http://www.wi-fi.org/knowledge_center/wpa2. Version: 2007 (Zitiert auf Seite 64)

[Wiko8a] Wikipedia: *Air Hockey*. http://de.wikipedia.org/w/index.php?title=Air_Hockey&stableid=49785732. Version: August 2008 (Zitiert auf Seite 35)

[Wiko8b] Wikipedia: *Bemani*. http://de.wikipedia.org/w/index.php?title=Bemani&stableid=53857148. Version: August 2008 (Zitiert auf Seite 56)

[Wiko8c] Wikipedia: *Boktai*. http://de.wikipedia.org/w/index.php?title=Boktai&stableid=52936711. Version: June 2008 (Zitiert auf Seite 31)

[Wiko8d] Wikipedia: *Browser game*. http://en.wikipedia.org/w/index.php?title=Browser_game&oldid=260501471. Version: August 2008 (Zitiert auf Seite 52)

[Wiko8e] Wikipedia: *Browserspiel*. http://de.wikipedia.org/w/index.php?title=Browserspiel&stableid=52086563. Version: August 2008 (Zitiert auf den Seiten 24, 52 und 61)

[Wiko8f] Wikipedia: *Comunio*. http://de.wikipedia.org/w/index.php?title=Comunio&stableid=53293166. Version: August 2008 (Zitiert auf Seite 55)

[Wiko8g] Wikipedia: *Enterprise JavaBeans*. http://de.wikipedia.org/w/index.php?title=Enterprise_JavaBeans&stableid=54154062. Version: August 2008 (Zitiert auf Seite 70)

[Wiko8h] Wikipedia: *Quake Live*. http://en.wikipedia.org/w/index.php?title=Quake_Live&oldid=228572208. Version: July 2008 (Zitiert auf Seite 61)

[Wiko8i] Wikipedia: *Travian*. http://de.wikipedia.org/w/index.php?title=Travian&stableid=49719002. Version: August 2008 (Zitiert auf Seite 58)

[Wiko8j] Wikipedia: *Yetisports*. http://de.wikipedia.org/w/index.php?title=Yetisports&stableid=47786069. Version: June 2008 (Zitiert auf Seite 60)

[Wino3] Winter, David: *Noughts And Crosses - The oldest graphical computer game*. Auf den Seiten von PONG-Story. http://www.pong-story.com/1952.htm. Version: April 2003 (Zitiert auf Seite 108)

[WKB+03] Wittenbrink, Heinz ; Köhler, Werner ; Bergmann, Olaf ; Jung, Bernhard ; Witt, Andreas ; Sasaki, Felix ; Lenz, Eva-Anna ; Trippel, Thorsten ; Milde, Jan-Thorsten ; Poenninghaus, Jens ; Köhler, Werner (Hrsg.) ; Wittenbrink, Heinz (Hrsg.): *XML*. Berlin : TELES European Internet Academy AG i.A. der TEIA Lehrbuch Verlag GmbH, 2003 (TEIA-Lehrbuchreihe). – 16–784 S. (Zitiert auf Seite 71)

[ZGL03] Zeimpekis, Vasileios ; Giaglis, George M. ; Lekakos, George: A taxonomy of indoor and outdoor positioning techniques for mobile location services. In: *SIGecom Exch.* 3 (2003), Nr. 4, S. 19–27. http://dx.doi.org/http://doi.acm.org/10.1145/844351.844355. – DOI http://doi.acm.org/10.1145/844351.844355 (Zitiert auf Seite 65)

[Zhao6] Zhang, Haiyan: *Control Freaks*, Interaction Design Institute Ivrea, Thesis Report, May 2006. http://failedrobot.com/thesis/ (Zitiert auf Seite 34)

[Zif07] Ziff Davis Enterprise Holdings Inc.: *Nokia unveils Linux-powered N810 Internet Tablet*. Auf den Seiten von LinuxDevices.com. http://www.linuxdevices.com/news/NS3669465936.html. Version: October 2007 (Zitiert auf Seite 18)

Alle URLs wurden zuletzt am 11. Januar 2009 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Christoph Stach)