Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht

Dennis Dedaj und Thomas Preisler

Pervasive Gaming Framework
The World Within

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Dennis Dedaj und Thomas Preisler

Pervasive Gaming Framework
The World Within

Ausarbeitung im Rahmen des Projektes
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Prüfer : Prof. Dr. Olaf Zukunft

Abgegeben am 6. März 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Dieser Projektbericht dokumentiert das Masterprojekt „Pervasive Gaming Framework“ der HAW Hamburg im Wintersemester 2008/2009. Im Folgenden wird die Motivation des Projektes und die Gliederung der Arbeit erläutert.

1.1. Motivation

Mobile Endgeräte, wie Mobiltelefone oder PDAs wurden in den vergangenen Jahren immer leistungsstärker und bieten mit GPS, Bluetooth und WLAN immer mehr interessante Funktionen. Mit der steigenden Verbreitung von leistungs- und funktionsstarken mobilen Endgeräten, gewinnen auch pervasive Spiele immer mehr an Bedeutung.

> *„Pervasive Games sind ein neues Spielformat, das die reale Umgebung mit virtuellen Elementen verbindet und dabei andere Aktivitäten zeitlich, räumlich und sozial durchdringt.“* Prinz (2006)

Im Rahmen des Projekts im Masterstudiengang Informatik an der HAW Hamburg schlossen sich im Wintersemester 2008/2009 acht Studenten unter der Leitung von Prof. Dr. Olaf Zukunft zusammen, um unter dem Titel „*Pervasive Gaming Framework*“ ein Framework für solche pervasiven Spiele zu entwickeln. Das Interesse beider Autoren an sogenannten Massive Multiplayer Online Role Play Games (MMORPG) führte zu der Idee, ein pervasives MMORPG als exemplarische Anwendung für das Framework zu entwickeln.

1.2. Gliederung der Arbeit

Im Rahmen dieser Ausarbeitung werden die beiden Autoren das Projekt aus der Sicht der Anwendung „*The World Within*“ beschreiben.
In Kapitel 2 wird zunächst das Projekt genauer beschrieben, es werden die verwendeten Technologien ebenso vorgestellt wie die Projektteilnehmer und die Aufgabenverteilung. Am

Ende von Kapitel 2 wird die eigene Zielsetzung beschrieben. In Kapitel 3 werden die fachlichen und technischen Anforderungen an die entwickelte Anwendung erhoben, bevor in Kapitel 4 das Design der Anwendung beschrieben wird. Kapitel 5 gibt Auskunft über die Realisierung der Anwendung, hier wird ein Überblick über die entwickelte Anwendung gegeben, wobei auf einige Implementierungsdetails genauer eingegangen wird. Zum Schluss wird in Kapitel 6 ein Fazit für die Anwendung sowie das Projekt gezogen und es wird ein Ausblick über mögliche Weiterentwicklungen gegeben.

2. Projektbeschreibung

Das Projekt hatte eine Dauer von einem Semester, bei einem Umfang von 8 Semesterwochenstunden und war im dritten Semester (Wintersemester 2008/2009) des Masterstudiengangs Informatik angesiedelt. Das Projektteam bestand aus 8 Studenten und wurde von Prof. Dr. Olaf Zukunft betreut.
Im Folgenden werden zunächst kurz die verwendeten Technologien beschrieben, bevor auf die Teamzusammenstellung und die Aufgabenverteilung sowie die Zielsetzung des Projektes, speziell die der beiden Autoren, eingegangen wird.

2.1. Verwendete Technologien

Dieses Kapitel gibt einen kurzen Überblick über die verwendeten Technologien.

2.1.1. Endgerät

Als Plattform für das mobile Endgerät gab es zum Projektstart drei Alternativen: Android, iPhone und Windows Mobile.
Obwohl alle Projektteilnehmer die Verwendung von Android favorisierten gab es zunächst kritische Stimmen, die auf die Gefahr hinwiesen, dass es zum Projektstart noch keine Android–Endgeräte gab und es somit nicht sicher sei, dass im Laufe des Projektes ein Endgerät verfügbar sein würde. Nach einigen kritischen Diskussionen entschied sich die Projektgruppe dann aber einstimmig für die Verwendung von Android, da man sich viel von dieser neuen Plattform versprach. Für die Verwendung von Android sprachen eine hohe Motivation bei den Projektteilnehmern, die Möglichkeit in Java zu entwickeln, sowie ein weitgehend ausgereifter Emulator. Gegen eine Verwendung des iPhones sprach eine den Projektteilnehmer unbekannte Programmiersprache (Objective C), was als zusätzliches Risiko angesehen wurde, sowie hohe Anschaffungskosten, da die von Apple bereitgestellte SDK nur auf neuen Mac-Computern installiert werden kann. Gegen Windows Mobile wurde sich aufgrund der geringen Motivation der Projektteilnehmer für diese Plattform entschieden.
Das erste erhältliche Androidphone ist das T-Mobile G1[1]. Ein solches Endgerät konnte nach

[1] siehe http://www.t-mobile.de/g1 Zugriff am: 1. März 2009

etwa der Hälfte des Projektes besorgt werden.
Im Folgenden wird die Android–Plattform kurz vorgestellt.

Android

Android ist ein Software–Stack für mobile Geräte der aus Betriebssystem, Middleware und Anwendungen besteht. Dabei bildet Android eine Abstraktionsschicht zur eigentlichen Geräte–Hardware. Android wurde zum größten Teil in Java entwickelt und erlaubt Anwendungsentwicklern die Verwendung einer Hochsprache (Java).

2.1.2. Server

Die Auswahl der serverseitigen Technologie wurde einstimmig von allen Projektteilnehmern entschlossen. Durch die Möglichkeit die Programmiersprache für die Serverkomponente frei wählen zu können, fiel diese Entscheidung zu Gunsten der Programmiersprache Java, da diese allen Projektteilnehmern bekannt ist.

2.1.3. Entwicklungswerkzeug

Bei der Wahl des Entwicklungswerkzeuges wurde den Empfehlungen von Google[2] gefolgt. Das Eclipse Plugin „Android Development Tools (ADT)“ bietet die vollständige Integration des Android Emulators, nützliche Editoren für Android GUIs und XML-Definitionen. Weiterhin wird das Deployen und Debuggen von Anwendungen auf dem T-Mobile G1 unterstützt.

2.2. Teamaufbau und Aufgabenverteilung

Das Projektteam setzte sich zusammen aus Amine El-Ayadi, Arazm Hosieny, Tobias Hutzler, Sascha Kluth, Julia Plischka, Peter Salchow und den beiden Autoren dieses Berichtes, Dennis Dedaj und Thomas Preisler. Die Projektleitung wurde von Julia Pliszka übernommen, mit Arazm Hosieny als Vertreter.
Schon in der ersten Findungsphase, während den Semesterferien vor dem offiziellen Projektstart, hatten die beiden Autoren die Idee ein pervasives Rollenspiel zu entwickeln. Das erste Ziel war es die Elemente eines klassischen Rollenspiels aufzugreifen und zu überlegen wie man diese möglichst pervasiv umsetzen kann. Eine genauere Beschreibung der eigenen

[2] siehe http://code.google.com/intl/de-DE/android/documentation.html Zugriff am 1.März 2009

Zielsetzung erfolgt im nächsten Abschnitt. Innerhalb der ersten Projekttreffen konnte auch Sascha Kluth für die Idee begeistert werden, sodass das Team „Rollenspiel“ komplett war.

Amine El-Ayadi und Julia Plischka schlossen sich zum Team „iLife“ zusammen, ihr Ziel war es eine Kalenderanwendung zu entwickeln, um mit Bekannten wichtige Termine austauschen zu können. (El-Ayadi (2009) u. Pliszka (2009))

Arazm Hosieny, Tobias Hutzler und Peter Salchow entschieden sich das eigentliche Framework zu entwickeln, welches von den beiden Anwendungen benutzt werden sollte. Ihre Aufgabe war es, aus den Anforderungen der beiden anderen Gruppen die Gemeinsamkeiten zu erkennen und diese im Framework zu implementieren (vgl. Hosieny (2009), Hutzler (2009) u. Salchow (2009)).

2.2.1. Aufgabenverteilung Team „Rollenspiel“

Innerhalb des Teams „Rollenspiel“ kristallisierte sich schnell folgende Aufgabenverteilung heraus:
Sascha Kluth wollte sich mit der Anbindung externer Hardware an den Android–Emulator beschäftigen, da zunächst davon ausgegangen werden musste, dass während der Projektlaufzeit kein Endgerät zur Verfügung steht. Mit Hilfe der externen Hardware sollten die bekannten Funktionen des Endgerätes auch im Emulator nutzbar sein (Kompass, Bewegungssensoren, (Video-)Kamera und GPS Modul). Außerdem wollte Sascha Kluth sich mit der Anbindung eines Wii–Controllers beschäftigen, vor dem Hintergrund der Metapher „Angriff mit einem Schwert“.
Die Autoren dieser Arbeit haben viele Aspekte wie Regelwerk und Spielmechanik gemeinsam erarbeitet, sodass eine genaue Trennung der Aufgaben bei diesen Aspekten nicht möglich ist. Auch in den Bereichen Postioning und Kommunikation gab es viele gemeinsame Überlegungen. Diese Bereiche wurden aber dahin gehend aufgeteilt, dass Dennis Dedaj sich hauptverantwortlich um die Positioning–Dienste kümmerte und Thomas Preisler für die Kommunikation hauptverantwortlich war.

2.3. Zielsetzung

Die Zielsetzung der Autoren für das Projekt war die Entwicklung eines pervasiven Rollenspiels unter Nutzung eines gemeinsam verwendeten Frameworks.
Die Idee entstand aus der Verknüpfung von bekannten pervasiven Anwendungen wie beispielsweise der „PS Community Manager“, der in einem vorhergehenden Projekt (vgl. Hartmann und Schönherr (2008), Dreyer (2008), Kruse (2008) u.a.) entwickelt wurde, und be-

kannten MMORPG wie beispielsweise „World of Warcraft“[3].
Anwendungen wie der „PS Community Manager“ folgen dem pervasiven Charakter sogenannter „Friend Finder“ Anwendungen. Durch das Auftauchen von GPS-fähigen Endgeräten und mobilen Internetflatrates wird die Kommunikation und die Positionierung allgegenwärtig ermöglicht. Aus diesem Grund wird es möglich, andere Personen zu lokalisieren und mit diesen zu kommunizieren.
Bei MMORPGs geht es darum, seinen Avatar zu trainieren, mit Gegenständen auszurüsten, mit anderen Avataren handeln zu lassen und gegen andere Avatare kämpfen zu lassen, um schließlich ein motivierendes Spielerlebnis zu erleben.
Durch die Verknüpfung beider Prinzipien werden interessante Möglichkeiten eröffnet. Insbesondere der Ort des Spielens sollte hier hervorgehoben werden, da viele Spieler in regulären MMORPGs sozial interagieren, dies jedoch vornehmlich unpersönlich stattfindet. Durch das Entwickeln eines pervasiven MMORPGs könnten diese Spieler auch in der realen Welt zusammenkommen und aufregende Abenteuer erleben. Weiterhin könnten aufregende Momente entstehen, sobald ein Spieler einen anderen angreift ohne, dass der Angegriffene das im ersten Moment realisiert. Erst durch das Signalisieren eines Angriffs durch das Endgerät, kann der angegriffene Spieler auf die Situation reagieren, indem er z.B. flüchtet oder einen Gegenangriff durchführt.

[3]siehe http://www.wow-europe.com/de/index.xml. Zugriff am: 02.März 2009

3. Anforderungen

Im diese Kapitel werden die fachlichen und technischen Anforderungen an die entwickelte Anwendung erhoben. Die Aufteilung der fachlichen Anforderungen in Regelwerk und Spielmechanik sowie die der technischen Anforderungen in Positioning, Kommunikation und Persistenz wird dabei auch in den folgenden Kapiteln eingehalten.

3.1. Fachliche Anforderungen

Folgend werden die fachlichen Anforderungen an das Regelwerk und an die Spielmechanik erläutert. Aus diesen findet in Kapitel 4 der spätere Entwurf der Anwendung statt.

3.1.1. Regelwerk

Bevor mit dem Design eines pervasiven Rollenspiels begonnen werden konnte, mussten die Anforderungen an dieses Regelwerk beschrieben werden. Im Folgenden werden die durch die Autoren zusammengefassten Anforderungen erläutert.
Die wichtigste Komponente eines Rollenspiels ist der **Avatar**. Ein Avatar besitzt diverse Attribute und Werte, durch welche er bzw. sein derzeitiger Zustand repräsentiert wird. Im Folgenden sind die primären Attribute aufgezählt:

- Name
  Jeder Avatar hat einen Namen, der durch den Spieler festgelegt wird.
- Typ/Klasse
  Jeder Avatar gehört einer Klasse an. Für den Prototypen sind die aus den meisten Rollenspielen bekannten Klassen vorgesehen: Krieger, Magier und Schurke.
- Leben (Life)
  Repräsentiert die aktuelle Lebensenergie des Avatars. Sinkt diese auf 0, stirbt der Avatar. Die Lebensenergie kann durch das Trinken von speziellen Tränken wieder aufgefüllt werden.

- Energie (Energy)
  Jede Aktion oder Fähigkeit, die ein Avatar ausführt kann eine bestimmte Menge an Energie verbrauchen. Die Energie kann genau wie das Leben durch das Verwenden von Tränken aufgefüllt werden.
- Level
  Jeder Avatar hat eine bestimmte Stufe. Je höher die Stufe, desto stärker ist der Avatar.
- Erfahrung (Experience)
  Durch das Sammeln von Erfahrung, wird das Aufsteigen des Levels ermöglicht.
- Stärke (Strength), Geschicklichkeit (Dexterity) und Intelligenz (Intelligence)
  Diese Attribute dienen der Berechnung der Angriffs- und Verteidigungswerte.

Weiterführend hat jeder Avatar spezielle **Fähigkeiten (Skills)**, die den Charakter und die Klasse des Avatars weiter ausprägen sollen.

- Jeder Avatar soll die Möglichkeit haben, mehrere Fähigkeiten zu besitzen.
- Avatare sollen die Möglichkeit haben, weitere Fähigkeiten zu sammeln oder zu lernen.
- Jede Fähigkeit kann Kosten haben, die dem Avatar von seiner Energie abgezogen werden, sobald er die jeweilige Fähigkeit benutzt.

Avatare tragen eine **Ausrüstung (Equipment)**.

- Avatare können mehrere Ausrüstungsgegenstände anziehen.
- Für den Prototyp soll das Tragen von einer Waffe, einer Rüstung und eines Ringes realisiert werden.
- Das Anziehen von Ausrüstungsgegenständen beeinflusst die Attribute des Avatars.

Avatare haben die Möglichkeit weitere Gegenstände, Tränke oder Ausrüstungsteile in ihrem **Inventar (Inventory)** mit sich zu tragen und zu benutzen.

- Mehrere Gegenstände sollen aufgehoben und in dem Inventar getragen werden können.
- Ausrüstungsteile die sich in dem Inventar befinden, sollen angezogen werden können.
- Benutzbare Gegenstände sollen aus dem Inventar heraus, benutzt werden können.

Weiterhin wurden während der Anforderungserhebung konkrete Formeln für das Berechnen der Angriffs- und Verteidigungswerte definiert. Diese sind nicht balanciert worden, um faire Kämpfe zu ermöglichen. Die Erstellung der Formeln erforderte Geschick und Kenntnisse über die Rollenspieldomäne, über die beide Autoren verfügen. Dadurch konnte eine einfache Berechnung der Werte relativ schnell festgelegt werden. Die konkreten Formeln werden

hier nicht aufgelistet, da diese nicht den Fokus dieser Arbeit darstellen sollen. Bei Interesse an den Formeln sollte sich der Quelltext der Anwendung bzw. des Regelwerkes direkt angeschaut werden.

3.1.2. Spielmechanik

Nachdem grundlegenden Regelwerk des Rollenspiels hatten die Autoren, sich eine Reihe von fachlichen Anforderungen überlegt, die den pervasiven Charakter des Spieles unterstreichen sollten. Diese werden im Folgenden vorgestellt.

Duellkampf

- Zwei Spieler sollen in der Lage sein ihre Avatare gegeneinander kämpfen zu lassen, wenn sie sich zum gleichen Zeitpunkt am gleichen Ort bzw. in einer bestimmten Entfernung zueinander befinden.
- Es sollen Metaphern, wie das Schwingen eines Schwertes, verwendet werden um eine Attacke zu erkennen. D.h. eine Attacke wird nicht nur durch das Drücken eines Knopfes ausgelöst, sondern es soll möglich sein externe Hardware mit dem Gerät zu verbinden (Wii–Controller), die mit Bewegungssensoren ausgerüstet ist, um eine Bewegung, wie einen Schlag zu erkennen.
- Der Schaden soll sich abhängig von der Intensität des Schlages berechnen.
- Wenn ein Spieler aus dem Kampf flüchten möchte, kann er einfach weglaufen. Sobald er sich außerhalb eines definierten Radius um den Gegner herum bewegt hat, ist der Kampf beendet.
- Der Spieler kann durch besonders ruhiges Verhalten oder durch unbemerktes Annähern spezielle Angriffe ausführen. Bsp.: Ein Spieler sieht einen Anderen und fotografiert ihn heimlich, der fotografierte Spieler wird darüber informiert und hat eine bestimmte Zeit lang die Chance den Spieler ebenfalls zu fotografieren. Gelingt ihm dieses nicht, erleidet er Schaden.

Handel

- Zwei Spieler können nur miteinander handeln, wenn sie sich zum gleichen Zeitpunkt am gleichen Ort bzw. in einer bestimmten Entfernung zueinander befinden.

- Es gibt Läden, bei denen Spieler Gegenstände kaufen können, wenn sie sich an einem bestimmten Ort befinden.

Kampf gegen Nicht-Spieler-Charakter

- Auch hier muss sich der Spieler an dem Ort befinden, an dem sich der Computergegner befindet.
- Außerdem ist ein bewegliches Monster denkbar, dass erst durch den Spieler gefunden werden muss. Das Endgerät könnte vibrieren, weil das große Monster den Boden zum Erschüttern bringt. Der Spieler kann nur anhand der Stärke der Vibration feststellen, ob er sich dem Monster nähert.

Augmented Reality

- Der Live-Stream des Kamerabildes soll mit den Positionen von Avataren und virtuellen Computergegnern angereichert werden. Hierbei wird nicht beabsichtigt, die Position des Avatars mit der des Spielers zu überlagern, sondern es soll an einer definierten Position der Avatar erkennbar sein.

Aufgaben

- Teamaufgaben: Spieler müssen in Kooperation zur selben Zeit an definierten Orten (diese können auch unterschiedlich sein) Aktionen durchführen.
- Guter Spieler / Böser Spieler: Ein „Guter“ Spieler bekommt eine Aufgabe, für welche er zu bestimmter Zeit an einem bestimmten Ort ankommen muss. Ein „Böser“ Spieler muss zur selben Zeit versuchen, den „Guten“ Spieler von seiner Aufgabe abzuhalten.
- Reiseaufgaben: Ein Spieler bekommt eine Aufgabe, bei der er einer definierten Route folgen muss.

Ergonomie

- Das Spiel muss jederzeit pausierbar sein, etwa wenn ein Anruf auf dem mobilen Endgerät eingeht.
- Nachdem es pausiert wurde, muss es möglich sein das Spiel wieder fortzusetzen.

- Es muss möglich sein das Spiel uneingeschränkt auch in einer ruhigen Umgebung, in der der Ton deaktiviert werden muss, zu spielen.

3.2. Technische Anforderungen

Dieses Kapitel stellt die technischen Anforderungen an das Framework, bzw. an die Anwendung, dar.

3.2.1. Positioning

Sämtliche Clients müssen, auf Grund der pervasiven Anforderungen an das Spiel, stetig ihre Position bekannt geben. Das Aktualisieren der Positionsdaten sollte in festen Zeitintervallen und nach dem Zurücklegen einer Strecke durchgeführt werden. Dadurch sollte eine flüssige Darstellung der Positionen, auch wenn sich die Avatare sehr schnell oder sehr langsam bewegen, ermöglicht werden.
Weiterhin muss es den Clients ermöglicht werden, den Server nach sich in der Nähe befindlicher Spieler zu erkundigen. Dabei sollte ein Radius und eine Position vorgegeben werden, für die der Server die in diesem Bereich befindlichen Spieler zurück gibt.
Nach der fachlichen Anforderung an die Spielmechanik in Kapitel 3.1.2 ist das Angreifen nur möglich, wenn sich beide Spieler innerhalb einer maximalen Distanz zueinander befinden. Aus diesem Grund muss das Positioning ebenfalls die Funktion bieten, die Distanz zwischen zwei Punkten zu berechnen. Weiterhin muss mit jeder Angriff-Aktion die Position mit der aktuellen Position des Angegriffenen verglichen werden, um ein Flüchten aus dem Angriffsbereich des Attackierenden zu ermöglichen.

Durch die genannten technischen Anforderungen, ist es nötig, eine Liste mit allen Positionen der Spieler auf dem Server zu speichern und stetig zu aktualisieren.

3.2.2. Kommunikation

Aus den fachlichen Anforderungen im Bereich der Spielmechanik resultiert, dass die Spieler bzw. die Anwendungen auf den Endgeräten, in der Lage sein müssen untereinander bzw. mit einem zentralen Server zu kommunizieren. Kommunikation meint in diesem Fall den Austausch von beliebigen Daten.

Wenn zwei Spieler miteinander Handeln oder Kämpfen wollen, müssen dabei Daten zwischen den Spielern ausgetauscht werden. Das können die Gegenstände sein, die vom

einem Spieler zu anderen transferiert wurden oder auch Zahlenwerte, die ausdrücken wie viel Schaden eine Attacke gegen einen Spieler gemacht hat. Zusätzlich müssen diese Daten ebenfalls zum Server übertragen werden. Dabei muss sichergestellt sein, dass keine Daten verloren gehen bzw. ein Übertragungsfehler muss bemerkt und ggf. korrigiert werden können.

Ferner müssen die Avatardaten zwischen den einzelnen Clients und dem Server synchronisiert werden, es muss sichergestellt werden, dass es im System nur eine „Wahrheit“ über die Daten gibt. Die Daten sollen im gesamten System den gleichen Stand haben.

3.2.3. Persistenz

Die Avatare der Spieler und alle dazugehörigen Daten, wie z. B. die Items sollen einmal auf den mobilen Geräten der Spieler gespeichert werden und ebenfalls redundant auf einem zentralen Anwendungsserver. Dadurch soll den Spieler ermöglicht werden, ihre Avatare auch von unterschiedlichen Endgeräten spielen zu können, zusätzlich tritt der positive Nebeneffekt auf, dass die Daten der Avatare im Falle eines Datenverlustes auf dem Endgerät vom Server wiederhergestellt werden können.

- Avatare (und zugehörige Daten) sollen auf dem Endgerät und dem Server gespeichert werden können.
- Avatare (und zugehörige Daten) sollen wieder geladen werden können.

4. Design

Im Folgenden wird das Design der entwickelten Anwendungen beschrieben, dabei wird sich an der in Kapitel 3 erstellten Aufteilung der einzelnen Anwendungsteile orientiert.

Sämtliche Designentscheidungen mussten auf Grund des zeitlich beschränkten Projektes, der vielen möglichen Funktionen und der hohen Komplexität einiger Funktionen immer zwischen der Einfachheit des Prototypen und gleichzeitig hoher Flexibilität, Erweiterbarkeit, Änderbarkeit, Benutzbarkeit und der Robustheit getroffen werden.

4.1. Regelwerk

In Kapitel 3.1.1 wurden die Anforderungen an das Regelwerk aufgelistet. Auf Basis der Anforderungen wurde der Entwurf eines Klassendiagramms erstellt. Das Klassendiagramm ist in A.1 angehängt.
Die Attribute des **Avatars** konnten, meist ohne weitere Überlegungen, übernommen werden. Es musste nur der jeweilige Datentyp festgelegt werden.
Bei dem Erstellen des Klassendiagrammes wurden auch gleich die nötigen Methoden bestimmt. Während dem Entwerfen des Klassendiagramms hat sich mehrfach herausgestellt, dass während der Anforderungsanalyse elementare Teile übersehen wurden. So wurde z. B. die Methode `calcStrength():int` für die Berechnung der Stärke des Avatars erforderlich, weil die Stärke nicht allein über das Attribut des Avatars repräsentiert wird, sondern auch über eventuell getragene Ausrüstungsgegenstände mit einem zusätzlichen Effekt auf die Stärke. Servicemethoden wie diese sind in vielen Fällen nötig geworden.
Die modellierte Vererbung der konkreten Avatarklassen (Krieger, Magier und Schurke) von der abstrakten Klasse `Avatar` ist mit Blick auf die bekannten Vorteile der Generalisierung als selbstverständlich zu sehen.

Das **Inventar (Inventory)** ist als ein eigenes Objekt gekapselt, um spätere Erweiterungen zu ermöglichen. Das Inventar enthält eine Liste von Gegenständen (Items). Die Gegenstände sind wiederum einfache Javaobjekte, mit dem Zusatz, dass sie auch Gegenstandsanforderungen (ItemRequiements) haben können. Diese dienen beispielsweise dazu, dass ein schwacher Magier keine schwere Kettenrüstung tragen kann. Aus Zeitgründen wurden bisher keine Methoden zum Verwalten des Inventars implementiert.

Wie sich während des Designs herausstellte konnten die **Fähigkeiten (Skills)** äquivalent zu den Gegenständen (Items) modelliert werden, da auch diese von dem Avatar getragen, benutzt, gekauft und verkauft werden können. Selbst die Attribute sind bis auf das zusätzliche Attribut `energyCosts`, das die wie in Kapitel 3.1.1 beschriebenen Kosten für die Benutzung der Fähigkeiten darstellt, identisch. Aus diesen Gründen entschieden sich die Autoren für ein direktes Erben von der Klasse `Item`.

Die Modellierung der **Ausrüstung** führte mehrfach zu der Notwendigkeit kleinerer Redesigns. Insbesondere die Aufrufhierarchie, andere Komponenten oder Avatare sollten nur Servicemethoden an dem Avatar aufrufen, führte zu der mehrfachen Kapselung der `equip(aItem:Equipable)` Methode. Diese wird an dem Avatar aufgerufen, ruft darin wiederum die `equip(aItem:Equipable)` Methode der Ausrüstung auf, welche daraufhin unter der Bedingung, dass der Gegenstand überhaupt tragbar ist(erbt von dem Equipable Interface), an die jeweilige Methode `changeWeapon(aWeapon:Weapon)`, `changeArmor(aArmor:Armor)` oder `changeRing(aRing:Ring)` weiterleitet.

Den Autoren war es während des Designs ausgesprochen wichtig, die Flexibilität, Erweiterbarkeit und Einfachheit des Modells stetig zu gewährleisten. Diesem Anspruch konnte gerecht geworden werden.

4.2. Spielmechanik

In Abschnitt 3.1.2 Fachliche Anforderungen – Spielmechanik wurden eine Reihe von Anforderungen an das pervasive Rollenspiel aufgeführt. Aufgrund der relativ kurzen Projektlaufzeit war den Autoren dieser Arbeit klar, dass nicht alle gewünschten Anforderungen im Bereich der Spielmechanik umgesetzt werden können. Diese Einschränkungen wurden bereits in der Designphase berücksichtigt.
Zunächst wurde sich darüber verständigt, was die minimal zu erfüllenden Anforderungen im Bereich der Spielemechanik sind, um einen spielbaren Prototypen zu realisieren. Nach einigen Diskussionen innerhalb der Gruppe, wurde sich dafür entschieden den **Duellkampf** zwischen zwei Spielern für den Prototypen zu implementieren. Diese Entscheidung basierte hauptsächlich auf zwei Gründen:

1. Ein wichtiges Ziel war es, ein Spiel zu entwickeln das die direkte Interaktionen der Spieler untereinander fordert/fördert. Daher war es wichtig bereits im Prototypen eine Funktion zu implementieren, die es den Spieler ermöglicht mit anderen Spielern zusammen zu spielen.

2. Der Duellkampf eignete sich besonders gut um eine Reihe von pervasiven Features umzusetzen. Attacken können über alternative Aktionen ausgeführt werden (Metapher: Schwert), die Spieler können nur miteinander kämpfen, wenn sie sich in einer

bestimmten Entfernung zueinander befinden und die Positionen der Spieler können auf einer Karte angezeigt werden (location based services).

Abb. A.2 im Anhang zeigt das Design des Duellkampfes als Sequenzdiagramm. Zunächst muss sich ein Spieler vom Server eine Liste der anderen Spieler in einer bestimmten Entfernung zu sich holen. Aus dieser Liste kann der Spieler nun einen anderen Spieler als Gegner auswählen. In der Abbildung ist der Angreifer von Typ „Warrior“ und der Gegner vom Typ „Rogue“. Um den Schaden zu berechnen, welchen der Angreifer bei dem Gegner verursacht berechnet er zunächst seinen eigenen „Attack Value“, dann berechnet der Gegner seinen „Defense Value“, wenn der „Attack Value“ des Angreifers höher ist, als der „Defense Value“ des Gegners gelingt der Angriff. Nun berechnet der Angreifer seinen „Physical Damage“ und der Gegner seinen „Armor Value“. Wenn der „Physical Damage“ größer ist als der „Armor Value“ verursacht der Angreifer beim Gegner Schaden. Dieser Schaden wird dem Gegner von seinen Lebenspunkten abgezogen und der Angreifer wird über den Ausgang der Attacke informiert.
Wie die Nachrichten hierbei technisch ausgetauscht werden und wo genau die Berechnung stattfindet wird in Abschnitt 4.4 beschrieben.

Die Bereiche **Handel**, **Kampf gegen Nicht-Spieler-Charaktere**, **Augmented Reality**, **Aufgaben** und **Ergonomie** wurden für das Design des ersten Prototypen nicht berücksichtigt.

4.3. Positioning

Dieses Kapitel bezieht sich auf die im Abschnitt 3.2.1 genannten Anforderungen.
Der Ablauf des Positionings wurde wie in Sequenzdiagramm A.3 dargestellt entworfen. Im Folgenden soll der Ablauf kurz erläutert werden:

1. Die Clients (Spieler) müssen sich an dem Server angemeldet haben.
2. Sobald das Login erfolgreich durchgeführt wurde, fangen die Clients automatisch an (bzw. sobald Positionsdaten auf dem Endgerät vorhanden sind) die eigene aktuelle Position an den Server zu senden.
3. Der Server empfängt die Positionsdaten und speichert diese zusammen mit der eindeutigen Client ID[1] in einer Liste. Sollte für den entsprechenden Client bereits eine Position gespeichert sein, wird diese aktualisiert.

[1] Der Server kennt die eindeutige ID der Clients und kann diese dem richtigen Client zuordnen, da jeder Client einen eigenen bidirektionalen Kommunikationskanal erhält (vgl. Kapitel 4.4)

4. Sobald der Client `getNearbyPlayers()` aufruft und an den Server sendet, antwortet dieser mit der Liste aller in der Nähe befindlichen Spieler. Dabei wird die aktuelle Position des anfragenden Clients und eine Distanz[2] benutzt, um zu berechnen, welche der Spieler sich in der Nähe des Clients befinden.
5. Nachdem Empfangen der Positionen werden die Daten auf dem Endgerät gespeichert. Mit Hilfe des Observer Patterns ist es möglich bei Änderung der bekannten in der Nähe befindlichen Spieler den MapView oder andere Ansichten zu aktualisieren.
6. Der Spieler kann nun einen der Spieler auswählen und ihn attackieren.
7. Da die Berechnung wie in Kapitel 4.4 beschrieben, immer über den Server stattfindet, muss auch dort überprüft werden, ob sich die Clients nah genug aneinander befinden. Sollten sich die Spieler nah genug zueinander befinden, wird der Angriff auf dem Server berechnet.
8. Daraufhin werden die neuen (oder unveränderten) Zustände beider Spieler an beide Spieler zurückgesendet.

4.4. Kommunikation

Von zentraler Frage bei dem Design der Kommunikationsschicht war die Entscheidung, ob die Clients direkt mit einander kommunizieren können oder jegliche Kommunikation über den zentralen Server läuft. Dazu wurde zunächst an Hand des Designs für den Duellkampf analysiert welche Nachrichten ausgetauscht werden müssen und wer jeweils die Kommunikation initiiert.
Der erste Schritt ist eine Anfrage von Spieler A beim Server, welche anderen Spieler sich in seiner Umgebung befinden, d.h. initiiert wird die Verbindung vom entsprechenden Client. Damit der Server die Anfrage beantworten kann, braucht er die Information wo sich die Spieler befinden, daher müssen sich zunächst alle Clients beim Server anmelden und ihm jeweils ihre (GPS-)Positionen mitteilen. Beim Spielstart bauen also zunächst alle Clients eine Verbindung zum Server auf, um diesem ihre jeweilige Position mitteilen zu können. Zwischen den Clients und dem Server bestehen also schon jeweils Kommunikationskanäle. (Abb. A.4 verdeutlicht diesen Ablauf.)
Nachdem Spieler A vom Server nun die anderen Spieler in seiner Umgebung erhalten hat, kann er Spieler B angreifen, so dieser sich in seiner Umgebung befindet. Der Nachrichtenaustausch für die Attacke kann nun entweder zwischen den Clients direkt erfolgen oder über den Server laufen. Für den ersten Prototypen haben wir uns dafür entschieden die Kommunikation zentral über den Server laufen zu lassen. Das hatte zwei Gründe:

[2] Für den Prototyp wurde hier eine statische sehr hohe Entfernung gewählt.

1. Der Nachrichtenaustausch über einen zentralen Server ist einfacher zu implementieren, als der Aufbau eines Peer-to-Peer Netzes zwischen den Clients. Durch die Zeitersparnis konnten noch andere interessante Aspekte in der Projektzeit angegangen werden.
2. Wir hatten uns dafür entschieden, dass der Server die zentrale „Wahrheit“ hält, sprich die Avatare werden auf dem Server gespeichert. Daher hätten alle Änderungen an den Werten der Avatare auch dann dem Server mitteilt werden müssen, wenn die Kommunikation direkt zwischen den Clients stattgefunden hätte. Somit haben wir uns entschieden die Berechnung der Attacken auf dem Server vorzunehmen.

Der technische Ablauf im Fall einer Attacke sieht nun wie folgt aus, über den bidirektionalen Kommunikationskanal zwischen Client A und dem Server teilt Client A den Server mit, dass er Client B angreifen möchte. Auf dem Server findet die in Abschnitt 3.1.2 beschriebene Berechnung statt. Die Änderungen an den Avataren werden auf dem Server gespeichert und die Ergebnisse des Kampfes an Client A und B geschickt. Verdeutlicht wird dieser Ablauf in Abb. A.5.

Auch wenn die Kommunikationsschicht Teil des Frameworks ist, sind viele Überlegungen zur Kommunikationsschicht in Zusammenarbeit mit der Frameworkgruppe entstanden, daher wurde die Kommunikationsschicht an dieser Stelle nochmal aus Sicht der Anwendung beschrieben. Für weitere Details zur Kommunikationsschicht wird auf Salchow (2009) verwiesen.

4.5. Persistenz

Die Anforderungen im Bereich der Persistierung wurden, als für das Framework wichtige Anforderungen identifiziert und von der Frameworkgruppe bearbeitet (vgl. Hosieny (2009) u. Hutzler (2009)).

5. Realisierung

Gemäß der bereits bekannten Aufteilungen der einzelnen Anwendungsteile wird in diesem Kapitel auf die Realisierung der entwickelten Anwendung eingegangen. Warum diese um den Abschnitt „Augmented Reality“ erweitert wurde, wird im entsprechenden Abschnitt erklärt.

5.1. Regelwerk

Das Klassendiagramm aus Abbildung A.1 wurde so umgesetzt, wie es in der Abbildung dargestellt ist.

Die Umsetzung des Regelwerkes mit einem bestehenden Klassendiagramm war die geringste Herausforderung des Projektes, da hier keine technischen Probleme auftraten und die eigentliche Arbeit, das Entwerfen, bereits weitestgehend erledigt war. Das hat im Wesentlichen damit zu tun, dass es sich bei der Technologie um einfache Java Klassenhierarchien handelte. Diese werden durch beide Autoren beherrscht.

5.2. Spielmechanik

Der Duellkampf zwischen zwei Spielern wurde, wie in der Design–Phase entworfen umgesetzt. Zusätzlich wurden eine Reihe von Ansichten implementiert, die Informationen über den Avatar geben und die es dem Spieler erlauben die Ausrüstungsgegenstände des Avatars zu wechseln bzw. dessen Fähigkeiten auszuwählen. Der ungefähre Ablauf des Spieles im entwickelten Prototyp lässt sich wie folgt gliedern:

1. Der Spieler erstellt einen neuen Avatar und verbindet sich mit dem Server. (Das Laden von Avataren konnte aus zeitlichen Gründen nicht mehr implementiert werden, obwohl die Daten des Avatars auf dem Gerät gespeichert werden können (vgl. Hutzler (2009)).
2. Der Spieler kann die Eigenschaften seines Avatars ansehen und diese ggf. ändern.

3. Außerdem kann sich der Spieler die Ausrüstungsgegenstände des Avatars, alle Gegenstände die der Avatar bei sich trägt und die Fähigkeiten des Avatars ansehen.
4. Der Spiel kann nun in der Karten- oder der sogenannten Action-Ansicht gegen andere Spieler kämpfen. Wenn sich andere Spieler in seiner Nähe befinden, kann er einen davon auswählen und diesen attackieren. Wie im Design–Kapitel beschrieben, findet die eigentliche Berechnung dabei auf dem Server statt und der Spieler sowie der angegriffene Spieler werden über das Ergebnis informiert.
5. Zusätzlich kann sich der Spieler in einer Radar-Ansicht noch die Entfernung und Ausrichtung zu anderen Spielern anzeigen lassen.

5.3. Positioning

Die Positionierung wurde, wie in Kapitel 4.3 beschrieben, realisiert. Falls das folgende Projektteam sich auch für die Androidplattform entscheidet, soll zur vereinfachten Einarbeitung in das Positioning im Folgenden beschrieben werden, welche Schritte nötig sind:

1. Bevor der GPS-Service genutzt werden kann, muss in der 'AndroidManifest.xml' die Erlaubnis, auf den Positionierungsservice zuzugreifen, eingetragen werden:

   ```
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
   ```

   für 'feine' GPS-Positionierung und/oder

   ```
   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
   ```

   für die grobere Positionierung über die Mobilfunkzellen.
2. In der Activity wird ein LocationManager von den Android Systemservices geholt:

   ```
   this.myLocationManager =
           (LocationManager)getSystemService(Context.LOCATION_SERVICE);
   ```
3. Mit Hilfe des LocationManagers kann jederzeit auf die aktuelle Position (soweit das Gerät eine diese bestimmen kann) zugegriffen werden:

   ```
   this.myLocation =
           myLocationManager.getCurrentLocation("gps");
   ```
4. Außerdem kann mit Hilfe des Quellcodes B.1 ein automatisches Aktualisieren der Position mittels eines IntentReceivers verwendet werden.

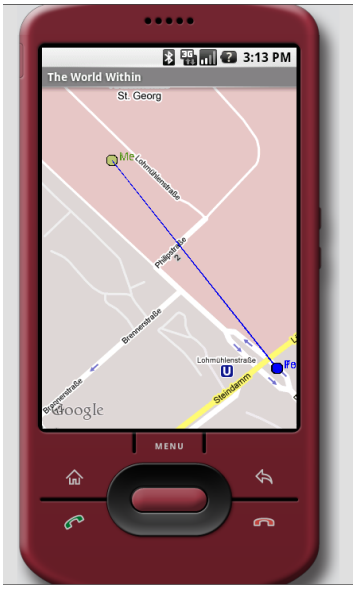

Abbildung 5.1.: Ansicht des implementierten MapViews

Eine Ansicht des erstellten MapViews (inkl. einem Overlay, auf dem die Spieler angezeigt werden) ist in Abbildung 5.1 ersichtlich.

Die Realisierung der Positionierung wurde wie oben beschrieben prototypisch implementiert, um Abhängigkeiten zwischen den Teams aufzulösen. Diese Implementierung wurde verworfen, nachdem das Frameworkteam eine eigene Lösung zur Verfügung stellte. Diese wird in Salchow (2009) beschrieben.

5.4. Kommunikation

Dieser Abschnitt erscheint an dieser Stelle nur der Vollständigkeit halber. Für Implementierungs–Details zur Kommunikationsschicht wird hier auf Salchow (2009) verwiesen.

5.5. Augmented Reality

Nachdem sich in der Design–Phase eigentlich dagegen entschieden wurde Augmented Reality–Funktionen in den ersten Prototypen zu integrieren, wurde diese Entscheidung in der Realisierungs–Phase sehr zum Ende des Projektes wieder verworfen.
Der Grund dafür war, dass innerhalb der Gruppe das Gefühl aufkam noch eine Funktion

implementieren zu wollen, die die Begeisterung der Anwender für das Spiel steigert. Die bereits angedachten Augmented Reality–Funktionen schienen hierfür genau richtig zu sein. Die Erweiterung der Realität wirkt auf viele Anwender faszinierend und ermöglicht dem Spiel sich, von anderen Spielen mit ähnlichen Inhalten abzusetzen.

Dadurch dass der Gruppe nur ein Endgerät zur Verfügung stand wurden zwei unterschiedliche Implementierungsansätze gewählt. Sascha Kluth entwickelte eine Lösung für den Emulator, der das Bild einer extern angeschlossenen Kamera erweitert (Kluth, 2009). Thomas Preisler entwickelte eine Lösung für das Endgerät, die im Folgenden beschrieben werden soll.

Das Ziel war, wie bereits in den Anforderungen beschrieben, den Live-Stream des Kamerabildes mit den Positionen von Avataren zu erweitern. Dem Spieler werden auf dem Kamerabild eine Repräsentation der Avatare der anderen Spieler anzeigt. Im Prototypen werden diese durch ein rotes Quadrat repräsentiert. Angedacht ist später eine detaillierter Darstellung, die die Eigenschaften der Avatare berücksichtigt. Abb. 5.2[1] zeigt die entsprechende Vision, abhängig von der GPS-Position und der Ausrichtung des Spielers, sowie der GPS-Position der anderen Spieler, werden deren Avatare auf dem Kamerabild des Spielers angezeigt. Der Spieler kann dadurch sehen wo sich andere Spiele befinden (auch in der Entfernung) und kann, an Hand des Kamerabildes, ungefähr deren Position bestimmen.

Das Kamerabild lässt sich in Android relativ einfach erweitern, dazu muss eine Klasse geschrieben werden, die von der Framework-Klasse *SurfaceView* erbt und das Interface *SurfaceHolder.Callback* implementiert. Diese Klasse kann nun wie jedes andere Android Widget verwendet werden und es können andere Widgets auf diesem platziert werden[2]. Das Kamerabild wurde nun zunächst um Daten vom Kompass erweitert, die Informationen über die Ausrichtung und die Lage des Gerätes geben. Zusätzlich wurden die Positionen der anderen Avatare mit einem roten Quadrat angezeigt. Dazu wurde die GPS-Position der anderen Avatare vom Server abgefragt und abhängig von der eignen GPS-Position und der Ausrichtung wurde berechnet, wo auf dem Kamerabild die roten Quadrate angezeigt werden müssen. Allerdings weist diese Berechnung noch zahlreiche Vereinfachungen auf, sodass die Darstellung noch sehr genau sind. Aber schon der erste Prototyp des Spieles kann so die interessanten Möglichkeiten aufzeigen, die noch erreicht werden können. Abb. 5.3 zeigt das erweitere Kamerabild im Android–Emulator. Das Schachbrettmuster im Hintergrund repräsentiert im Emulator das eigentliche Videobild. Da das Bild aus dem Emulator aufgenommen wurde, fehlt an dieser Stelle auch das rote Quadrat, welches andere Spieler repräsentiert.

[1] Bilder von http://www.enkin.net und http://www.wizards.com/
[2] Siehe: http://developer.android.com/guide/index.html

Abbildung 5.2.: Augmented Reality mit Computergegner (Entwurf)

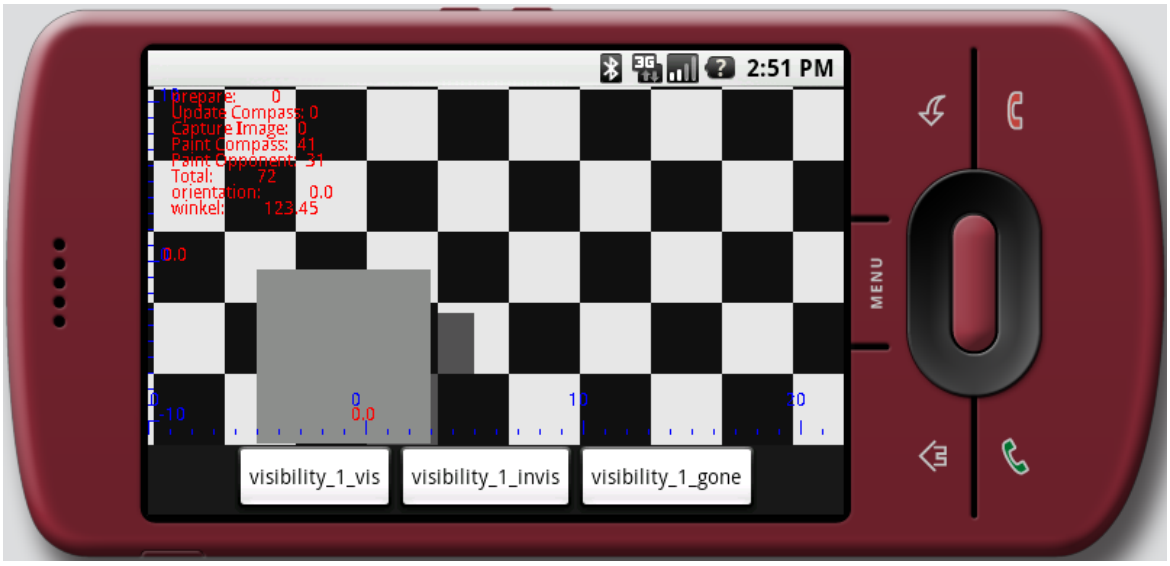

Abbildung 5.3.: Augmented Reality

6. Fazit und Ausblick

Abschließend wird ein kurzes Fazit über das Projekt gezogen und es wird ein Ausblick über mögliche Weiterentwicklungen gegeben.

6.1. Fazit

In diesem Abschnitt werden die Erfahrungen der Autoren in dem Masterprojekt zusammengefasst.

6.1.1. Anwendung

Für die Anwendung gab es viele interessante, konzeptionelle Ideen von denen aus zeitlichen Gründen leider nur wenige umgesetzt werden konnten. Trotzdem entstand am Ende eine interessante, spielbare Anwendung mit viel Potential für Weiterentwicklungen. Ferner konnten bei der Entwicklung interessante Ideen für mögliche Masterarbeitsthemen gewonnen werden.

Emulator

Obwohl der Android–Emulator eine vernünftige Basis für die Entwicklung bot, gab es besonders in Verbindung mit externer Hardware immer wieder Probleme mit dem Emulator. Beispielsweise ist die Umsetzung der per Hand eingegebenen Positionsaktualisierungen[1] ungenau. Diese eignet sich zum Testen nur sehr bedingt.

[1] Der Androidemulator bietet eine telnet Schnittstelle zum direkten simulieren von GPS-, GSM- und anderen Hardwarebefehlen

Hardware

Ein Problem bei der Entwicklung der Anwendung war, das lange Zeit kein Endgerät zur Verfügung stand, auf dem die Anwendung unter realistischen Bedingungen getestet werden konnte. Zum Ende des Projektes stand zwar ein Endgerät zur Verfügung, um eine Mehrspieler–Anwendung optimal testen zu können, wäre ein zweites Gerät aber unerlässlich gewesen.

Weitere Probleme entstanden durch die Verwendung von externer Hardware, die teilweise unter hohem Aufwand an den Emulator angeschlossen werden musste. Hier ging wertvolle Zeit verloren, die besser in die eigentliche Anwendung geflossen wäre. Ein Beispiel hierfür ist die Anbindung der GPS-Mäuse über Bluetooth an die Entwicklungs–Notebooks, die nur sporadisch funktionierte und gemäß Murphys Law[2] während der Präsentation auf dem Weihnachtsmarkt den Dienst verweigerte.

Trotz aller Probleme bleibt zu erwähnen, dass die Entscheidung, auf der Androidplattform zu entwickeln, nicht bereut wird. Android bietet eine durchdachte Architektur mit vielen Erleichterungen für Entwickler, auch wenn es noch Kinderkrankheiten hat.

6.1.2. Projekt

Obwohl das Ergebnis des Projektes durchaus als sehr positiv angesehen werden kann, muss doch rekapituliert werden, dass es im Projekt zu einigen vermeidbaren Fehler kam. Im Bereich der Projektplanung hätten alle Teilnehmer gewissenhafter Arbeiten müssen und Probleme hätten früher angesprochen werden müssen.

Die Vision „Ein Framework für pervasive Anwendungen zu entwickeln“ wurde während des Projektes aus den Augen verloren. Die Anwendungen wurden entwickelt, ohne den Frameworkaspekt stetig zu fokussieren.

Die Einteilung der Teams erwies sich meistens als vorteilhaft. Jedoch kam es auch zu Situationen, in denen es wünschenswert gewesen wäre direkte Änderungen an fremden Code durchführen zu können. Das ist jedoch nur möglich, wenn jeder den gesamten Quellcode kennt oder das Projektteam stetig in einem Projektraum gemeinsam entwickeln kann. Der zweite Vorschlag wäre hier vorzuziehen gewesen, da es eine gemeinsame Projektzeit gab und ein Projektraum in der HAW zur Verfügung stand[3]. Weiterhin sollte darauf geachtet werden, dass möglichst immer alle Projektteilnehmer vor Ort sind. Auch wenn es oft als nicht nötig erschien, traten oft spontan Probleme auf und diese hätten möglichst sofort gelöst werden sollen. Dadurch ließen sich weitere Abhängigkeiten lösen.

[2] „Whatever can go wrong, will go wrong.“
[3] Der Projektraum stand meistens zur Verfügung. Noch besser wäre ein dedizierter Raum gewesen.

Abschließend kann gesagt werden, dass die „üblichen“ Fehler im Software–Engineering gemacht wurden. Es wurde zu wenig Zeit auf die Projektplanung verwendet, die erstellten Pläne waren nicht fein granular genug und die Design–Phase wurde zu schnell abgeschlossen, sodass einige konzeptionelle Fehler erst während der Implementierung auffielen. Auch wurde die Anwendung aus zeitlichen Gründen nicht ausreichend getestet.

Positiv lässt sich vermerken, dass das Projekt trotzdem ein erfolgreiches Ergebnis hatte und aus den Fehlern viel gelernt werden konnte, was in nächsten Projekten (z. B. der Masterthesis) besser gemacht werden kann.

6.2. Ausblick

Dieses Kapitel gibt einen Ausblick auf die zukünftige Entwicklung der Rollenspielanwendung und die potenziellen Masterarbeiten der Autoren.

6.2.1. Redesign

Die Autoren dieser Arbeit haben sich vorgenommen auf Basis des ersten Prototypen ein komplettes Redesign der Anwendung vorzunehmen. Dabei wird es vornehmlich um zwei Aspekte gehen:

1. Aufgrund der geringen Erfahrung im Umgang mit der Android–Plattform zu Beginn des Projektes, wurde teilweise entgegen der Programmierparadigmen für Android implementiert. An diesen Stellen wäre eine Neuimplementierung sinnvoll.
2. Das allgemeine Ziel des Projektes war es ein Framework für pervasive Anwendungen zu entwickeln, sowie zwei Anwendungen, die auf diesem aufsetzen. Nun hat sich im Laufe des Projektes ergeben, dass nur eine der beiden Anwendungen das Framework nutzt. An einigen Stellen gilt es nun zu überlegen, ob die lose Kopplung zwischen Anwendung und Framework wirklich sinnvoll ist oder ob durch eine engere Kopplung sinnvolle Vereinfachungen erreicht werden könnten.

6.2.2. Künstliche Intelligenz

Eine Überlegung ist es, nach einem erfolgreichen Redesign der Anwendung, im Rahmen einer Masterthesis die Anwendung um, sich in der realen Welt bewegenden, künstlichen Intelligenzen zu weitern. Die Idee für die Masterthesis ist ein Framework für pervasive Spiele zu entwickeln, mit dem sich solche um künstliche Intelligenz erweitern lassen. Die künstliche

Intelligenz soll dabei in der Lage sein sich, wie die menschlichen Spieler, durch die reale Welt zu bewegen. Angedacht ist es nun die Verwendung eines solchen Frameworks an Hand der hier entwickelten Anwendung zu verdeutlichen. Die Anwendung würde dabei um Nicht-Spieler-Charaktere erweitert, die sich virtuell durch die reale Welt bewegen und mit denen die Spieler interagieren können (vgl. Preisler (2009)).

6.2.3. Content Generation

Bisher ist die virtuelle Welt des Rollenspiels fast unbevölkert. Spieler können ihre Avatare zwar bereits durch die Welt bewegen, treffen dabei jedoch nur auf andere Avatare. Für die Spielerfahrung ist das unzulänglich, da sich gerade zu Beginn eines solchen Spiels nicht genügend Spieler in der Nähe befinden. Die Idee für die Masterarbeit ist ein Autorenwerkzeug für die Generierung und Komposition von Inhalten für pervasiven Anwendungen. So könnte perspektivisch ein Werkzeug entwickelt werden, mit dem ein Spieledesigner Inhalte beschrieben kann und diese durch ein Softwaresystem instanziiert werden. Um das Interesse an der Seminar Ringvorlesung Ausarbeitung **?** zu wecken, wird hier auf die mögliche Beschreibung von Spieleraufgaben[4] durch ein Storyboard[5] indem abstrakte Entitäten, welche erst durch Ontologien an Semantik gewinnen und mit realen Entitäten der gleichen Semantik verknüpft werden, in einen zeitlichen Ablauf gebracht werden können. Durch dieses Vorgehen könnten Spielererlebnisse, unabhängig von Zeit und Ort, für pervasive Spiele definiert werden.

[4] Im Rollenspielgenre sogenannte Quests.
[5] Eine für Spieledesigner bekannte Darstellung.

A. Abbildungen

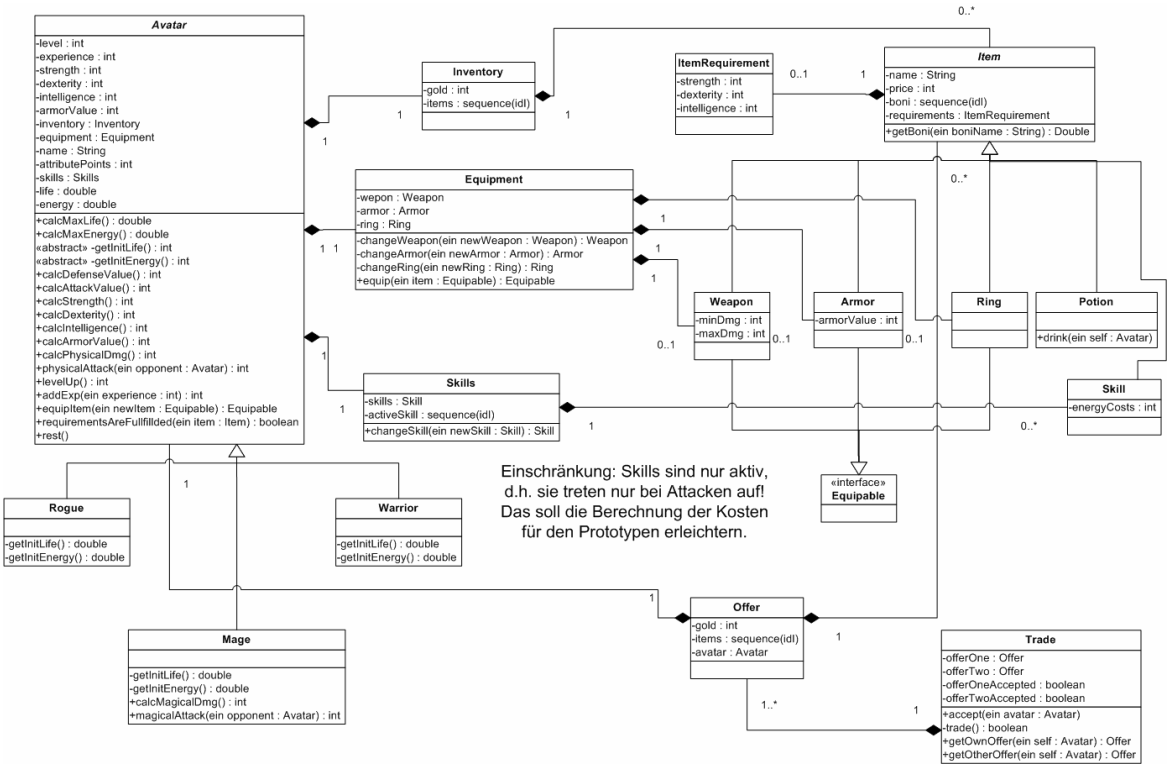

Abbildung A.1.: Das entworfene Klassendiagramm des Regelwerkes

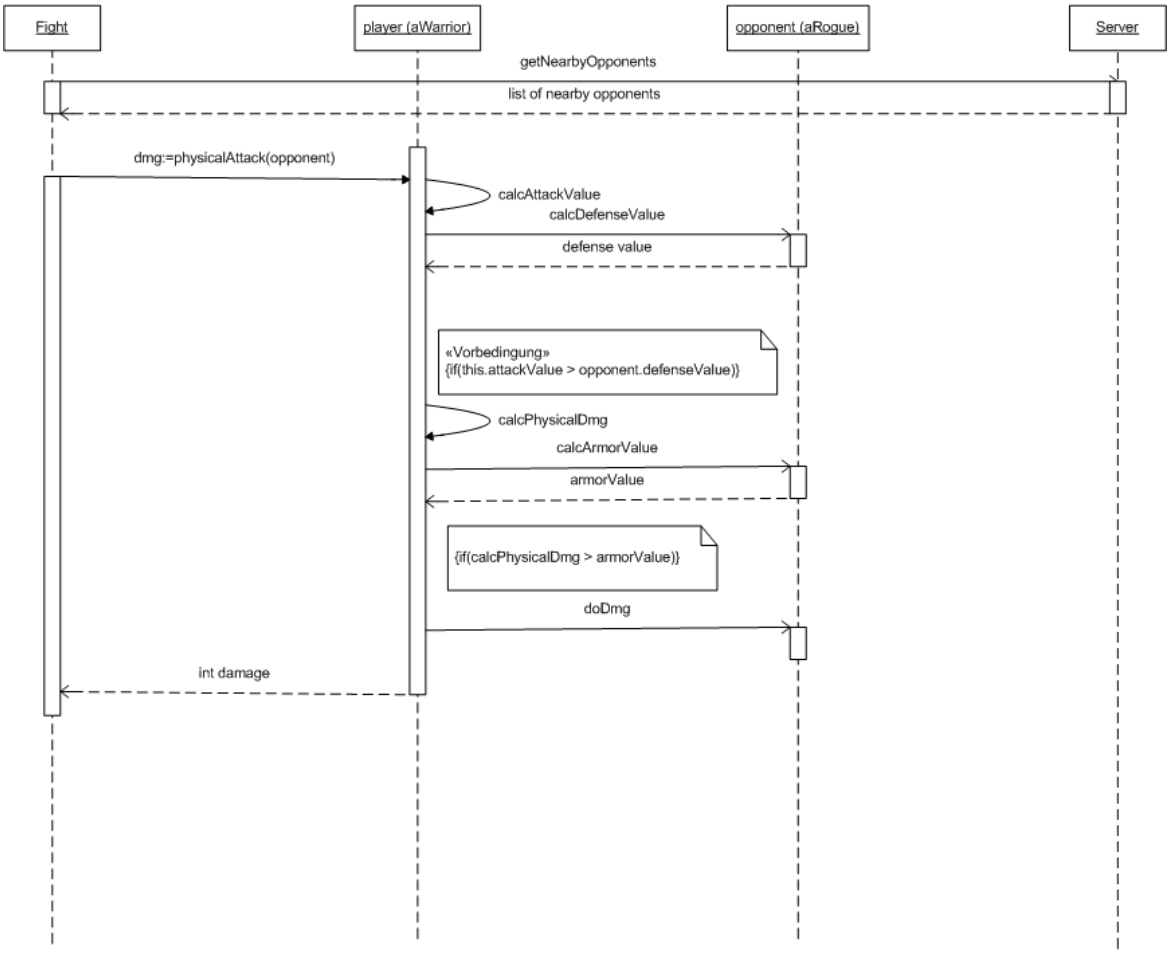

Abbildung A.2.: Sequenzdiagramm Duellkampf

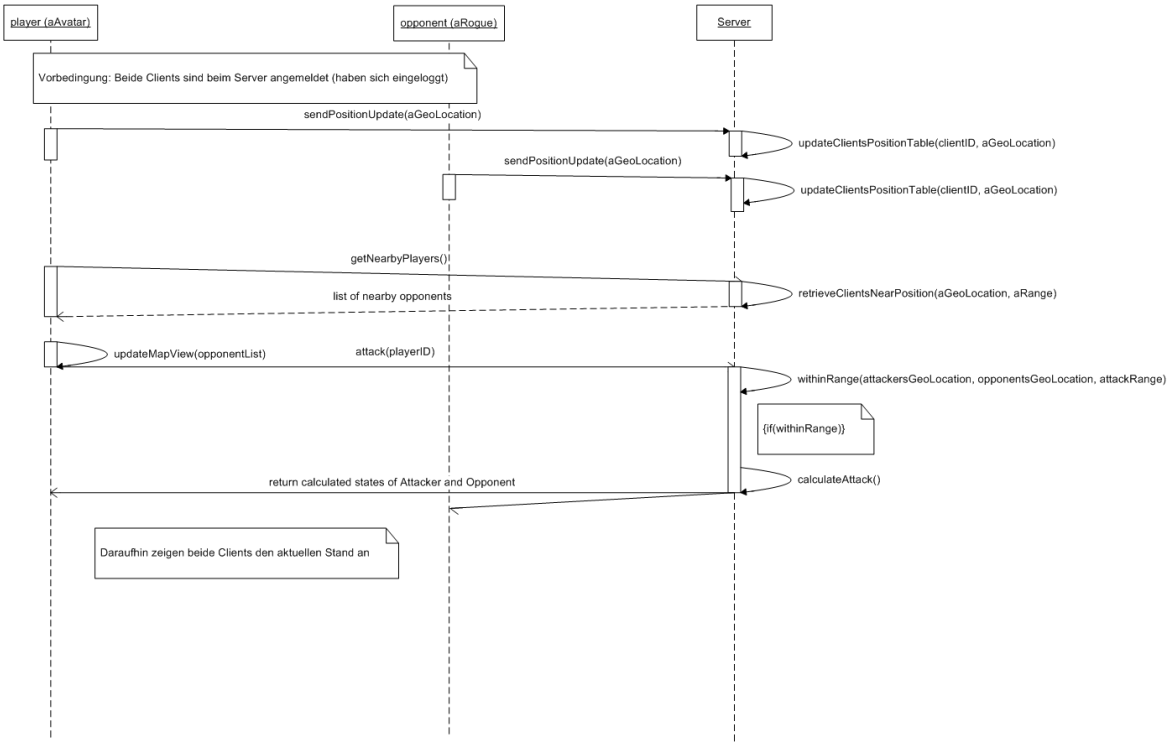

Abbildung A.3.: Sequenzdiagramm des Positionings

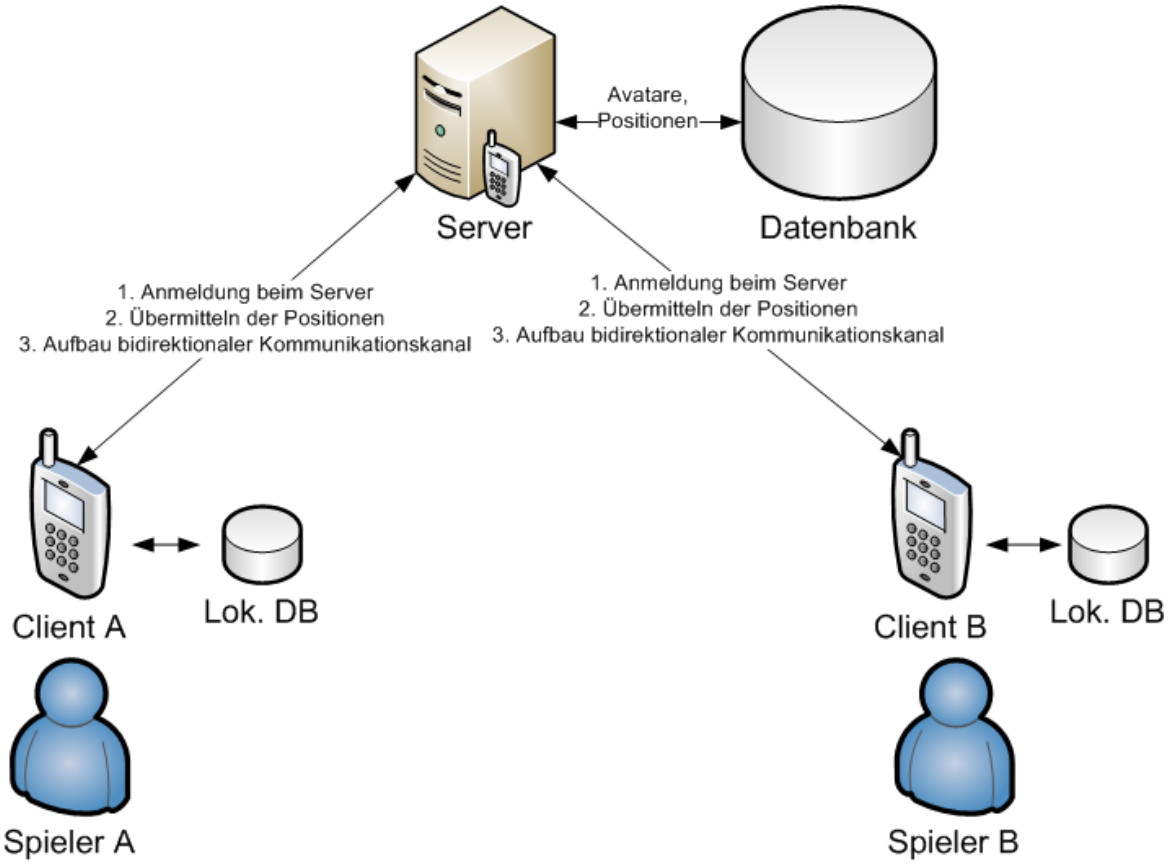

Abbildung A.4.: Initiale Kommunikation

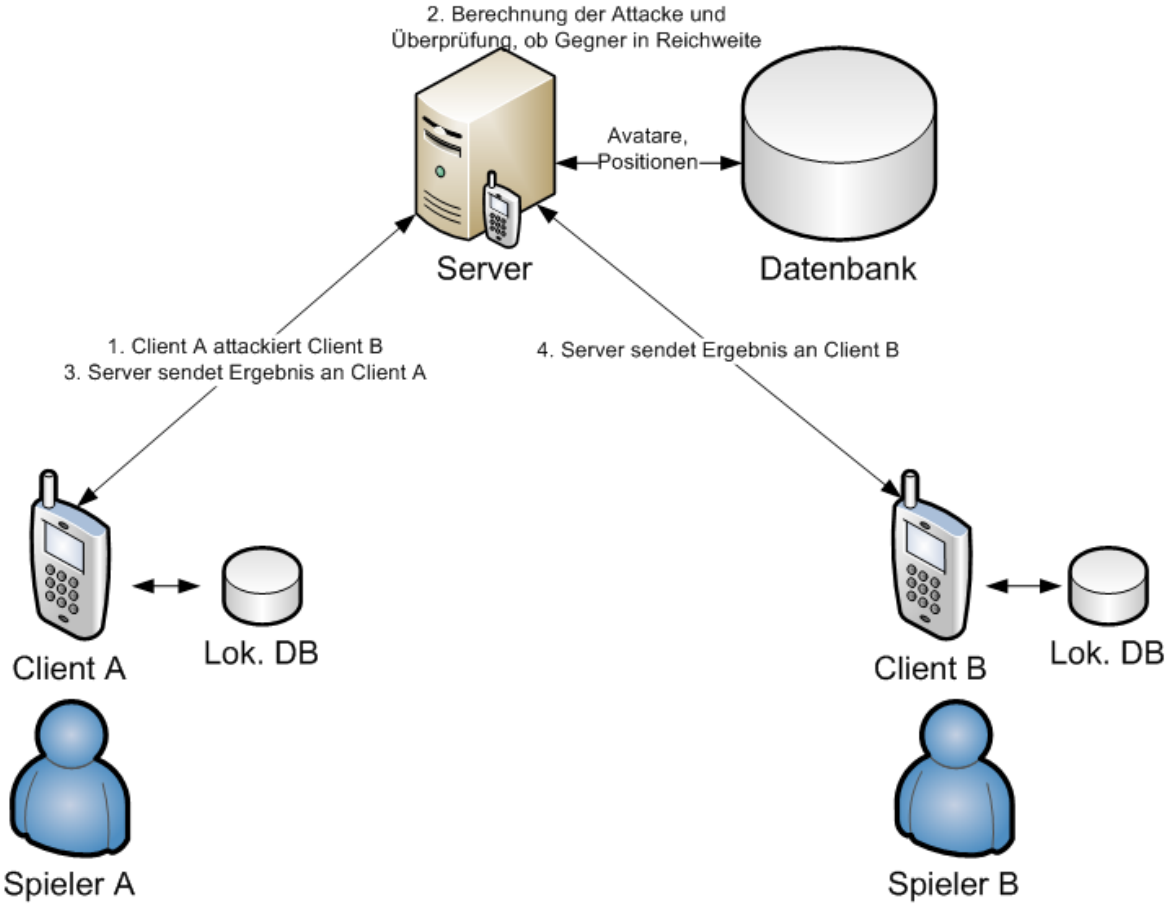

Abbildung A.5.: Kommunikation während einer Attacke

B. Listings

```
1
2     protected final IntentFilter myIntentFilter =  new IntentFilter(
          MY_POSITION_CHANGED_ACTION);
3
4     private void autoPositionConfiguration() {
5          // Den ersten verfügbaren Positions Provider holen.
6           List<LocationProvider> providers = this.myLocationManager.getProviders();
7           LocationProvider provider = providers.get(0);
8
9           this.myLocationManager.requestUpdates(provider, TIME_BETWEEN_UPDATE,
                DISTANCECHANGE_FOR_UPDATE, new Intent(MY_POSITION_CHANGED_ACTION));
10
11          this.myIntentReceiver = new MyIntentReceiver();
12    }
13
14
15     class MyIntentReceiver extends IntentReceiver {
16        @Override
17            public void onReceiveIntent(Context context, Intent intent) {
18                if(this.doUpdates){
19        //Aktualisiere deinen View oder ähnliches
20        ...
21        }
22          }
23
24    //Die folgenden Methoden sind unbedingt nötig, um Ressourcen zu sparen!
25    //Sauber an- und abmelden:
26
27     @Override
28     public void onResume() {
29          super.onResume();
30          this.doUpdates = true;
31          this.registerReceiver(this.myIntentReceiver, this.myIntentFilter);
32     }
33
34     @Override
35     public void onFreeze(Bundle icicle) {
36          this.doUpdates = false;
37          this.unregisterReceiver(this.myIntentReceiver);
38          super.onFreeze(icicle);
39     }
```

Listing B.1: Automatisches Aktualisieren von Positionen

Literaturverzeichnis

[Dedaj 2009] DEDAJ, Dennis: *Content Generation For Pervasive Games*. Seminararbeit Ringvorlesung, HAW Hamburg. 2009. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master08-09/vortraege.html

[Dreyer 2008] DREYER, Markus: *Bericht INF-M3 PO – Pervasive Spine*. Projektbericht, HAW Hamburg. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/dreyer/report.pdf

[El-Ayadi 2009] EL-AYADI, Amine: *Pervasive Gaming Framework*. Projektbericht, HAW Hamburg. 2009

[Hartmann und Schönherr 2008] HARTMANN, Leif ; SCHÖNHERR, Jan: *Bericht INF-M3 PO – Pervasive Spine Framework*. Projektbericht, HAW Hamburg. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/hartmann-schoenherr/report.pdf

[Hosieny 2009] HOSIENY, Arazm: *Pervasive Gaming Framework*. Projektbericht, HAW Hamburg. 2009

[Hutzler 2009] HUTZLER, Tobias: *Pervasive Gaming Framework*. Projektbericht, HAW Hamburg. 2009

[Kluth 2009] KLUTH, Sascha: *Pervasive Gaming Framework*. Projektbericht, HAW Hamburg. 2009

[Kruse 2008] KRUSE, Ralf: *Masterprojekt-Bericht – Pervasive Spine Framework*. Projektbericht, HAW Hamburg. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/kruse/report.pdf

[Pliszka 2009] PLISZKA, Julia: *Pervasive Gaming Framework*. Projektbericht, HAW Hamburg. 2009

[Preisler 2009] PREISLER, Thomas: *Virtuelle Agenten in der realen Welt - KI für pervasive Spiele*. Seminararbeit Ringvorlesung, HAW Hamburg. 2009

[Prinz 2006] PRINZ, Wolfgang: *Pervasive Games*. Fraunhofer FIT - IuK Wirtschaftssummit. 2006

[Salchow 2009] SALCHOW, Peter: *Pervasive Gaming Framework*. Projektbericht, HAW Hamburg. 2009