HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Jan Böttrich

Entwicklung eines verteilten Serversystems für ein Massive-Multiplayer-Online-Game

2011

Fakultät **Mathematik/Naturwissenschaften/Informatik**

BACHELORARBEIT

Entwicklung eines verteilten Serversystems für ein Massive-Multiplayer-Online-Game

Autor:
Jan Böttrich

Studiengang:
Informatik Bachelor

Seminargruppe:
IF06w2-B

Erstprüfer:
Prof. Dr.-Ing. Mario Geißler

Zweitprüfer:
Prof. Dr.-Ing. Uwe Schneider

Mittweida, August 2011

Bibliografische Angaben

Böttrich, Jan: Entwicklung eines verteilten Serversystems für ein Massive-Multiplayer-Online-Game, 45 Seiten, 14 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik/Naturwissenschaften/Informatik

Bachelorarbeit, 2011

Referat

Die vorliegende Arbeit beschäftigt sich mit der Erstellung eines Konzepts für ein Serversystem, das für vollkommen dynamische Onlinespielwelten Verwendung findet. Für das entwickelte Konzept ist eine minimale Beispielanwendung zu realisieren, anhand derer eine Praxistauglichkeit nachgewiesen werden kann. Für die Realisierung einer Spielwelt, welche zum Konzept passt, eignet sich ein Weltraumszenario.

Das zu konzipierende System muss in der Lage sein, eine Vielzahl von Clients zur gleichen Zeit zu bedienen. Bei der Konzeption steht dies im Mittelpunkt. Um schwankendem oder steigendem Ressourcenbedarf gerecht zu werden, soll das System im Betrieb durch zuschalten weiterer Komponenten oder Hardware erweiterbar sein.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Abkürzungsverzeichnis

API	Application Programming Interface; zu deutsch Programmierschnittstelle
DLL	Dynamic Link Library
GOaID	GameOrganisationAndInformationDistribution Komponente
LS	LoginServer Komponente
MMOG	Massive Multiplayer Online Game
MS	MasterServer Kompnente
MSDN	Microsoft Developer Network
MSDNAA	MSDN Academic Alliance
MySQL	OpenSource Datenbanksoftware von Oracle
RakNet	Netzwerkengine von Jenkinssoftware LLC
WD	WorldData Komponente
WP	WorldProcessing Komponente

1 Einleitung

Der Begriff des Massive Multiplayer Online Games, kurz MMOG, beschreibt eine Form von Spielen, bei denen möglichst viele Nutzer gleichzeitig mit einer Spielwelt interagieren. Spitzenreiter in diesem Segment betreiben Server, die zehntausende von Nutzern zur gleichen Zeit besuchen. Statistiken über die tatsächliche Frequentierung von MMOG Servern sind kaum veröffentlicht. Von der Isländischen Firma CCP sind allerdings Serverstatistiken online verfügbar[1]. Ihr Science Fiction Weltraum MMOG "Eve-Online" erreichte Zahlen von über 60.000 Spielern auf einer einzigen, zusammenhängenden Spielwelt zur gleichen Zeit.

In den letzten Jahren stieg die Zahl neuer Online-Spiele im Bereich MMOG zunehmend an[2]. Anhand der stetigen Verbesserung von Infrastruktur und Technik weltweit, ist auch weiterhin mit einem Wachstum zu rechnen. Die im Bereich der MMOGs als Marktführer zu bezeichnende Firma Blizzard Entertainment erzielt Benutzerzahlen von mehreren Millionen[3]. Derartig viele Nutzer erreicht sonst keiner. Diese Summe verteilt sich allerdings auf viele verschiedene Server und Welten. Des Weiteren sind die Spielwelten oft auch rein statisch, d.h. ein Großteil der virtuellen Umgebung ist unveränderbar, und die Einflussmöglichkeiten der Benutzer sind sehr eingeschränkt.

Die weitere Entwicklung geht klar in die Richtung dynamischer Spielwelten. Die Entwicklung eines Konzeptes für den Anwendungsfall einer Spielwelt, welche ausschließlich aus Objekten bestehen soll, die als dynamisch anzusehen sind, stellt das Ziel dieser Arbeit dar. Eine sinnvolle und nach Möglichkeit realistische Physikberechnung soll dabei ebenfalls berücksichtigt werden.

1.1 Problembeschreibung

Wie zuvor schon erwähnt, geht die Entwicklung im Bereich der MMOGs klar in Richtung von dynamischen, lebendigen, Spielwelten. Darüber hinaus stellt die Berechnung, Simulation und Nachbildung von Physik sehr hohe Anforderungen an Planung und Technik. Im Rahmen dieser Arbeit soll nun ein Serverkonzept für solch einen Anwendungsfall erarbeitet werden.

Eine weitere Entwicklung zielt verständlicherer Weise auf die Realisierung von belebte-

[1] vgl.: Statusmonitor für Eve-Online Server: <http://eve-offline.net/>, verfügbar am 10.08.2011

[2] vgl.: Handelsblattartikel zum Wachstum der Computerspielindustrie: <http://www.handelsblatt.com/markt-fuer-online-spiele-legt-kraeftig-zu/4484602.html>, verfügbar am 11.08.2011

[3] vgl.: Pressemitteilung von Blizzard Entertainment: <http://eu.blizzard.com/de-de/company/press/pressreleases.html?id=2450443>, verfügbar am 10.08.2011

ren Spielwelten hin. Die wenigsten Spieler erfreuen sich an immer gleichen Umgebungen oder Landschaften, welche ohne Eingriffe seitens der Betreiber auch nach Jahren genau das gleiche Erscheinungsbild beibehalten. Um diesen Umständen gerecht zu werden und vor allem dem Spieler mehr Möglichkeiten zu offerieren, bietet sich das Konzipieren von komplett dynamischen Systemen an. Simulation von Physik und die Fähigkeit, dass der Benutzer seine Umwelt verändern kann, sollen zu einer deutlich belebteren virtuellen Welt beitragen.

Für die Tatsache, dass bisher kaum dynamische Spielwelten in diesem Genre vertreten sind, gibt es vor allem einen Grund. Dieser liegt in der Tatsache, dass bei MMOGs möglichst wenig Daten zwischen Server und Client ausgetauscht werden sollen. Zum Einen wird versucht Netzwerklast, d.h. die Menge der zu übertragenden Daten und Nachrichten, so gut wie möglich zu vermeiden und zum Anderen soll die Last für den Server auch möglichst gering gehalten werden. Da sich aber die Netzwerk- bzw. Internetanbindung der meisten Haushalte in der letzten Jahren stetig verbessert hat, kann der Aspekt der Netzwerklast als weniger kritisch angesehen werden. Anders ausgedrückt kann eine höhere Datenübertragungsrate vorausgesetzt werden, genug um deutlich mehr als die für statische Spielwelten notwendige Datenvolumen zu bewältigen. In Folge dessen steigt die Notwendigkeit für ein effektives Serverkonzept, welches den Mehraufwand für eine dynamische Spielwelt bewältigen kann.

1.2 Zielsetzung der Arbeit

Mit dieser Arbeit soll ein Einblick in den Umgang mit dynamischen Spielwelten im Bereich von MMOGs gegeben werden. Die dabei wichtigen Einflussfaktoren werden genannt. Im Zentrum der Arbeit steht die Konzeption und Realisierung eines Serverkonzeptes für den genannten Anwendungsfall. Dazu ist der Anwendungsfall weiter auszuarbeiten und genauer zu beschreiben.

Mit dem Entwurf gilt es auch eine möglichst hohe Flexibilität in Bezug auf die gegebene Hardware zu ermöglichen. Dazu wird die Verbindung der einzelnen Serverkomponenten auf die Netzwerkschnittstelle minimiert und eine Parallelisierung der Komponenten in den Vordergrund gestellt. Alle Teile des Gesamtsystems sollten einer hierarchischen Struktur folgen und im Betrieb austauschbar sein. Ausnahmen dazu bilden nur die Datenbankschnittstelle und oberste Verwaltungseinheit, wobei letztere auch Redundanz aufweisen kann. Für den Client muss nur eine Vermittlungskomponente bekannt sein, welche ihn zu einer Bearbeitungskomponente weiterleitet.

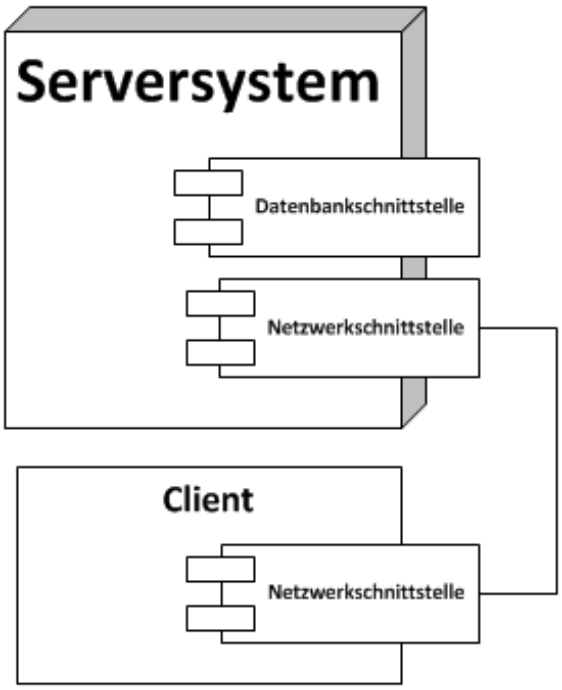

Abbildung 1.1: Externe Schnittstellen des Serversystem

1.3 Struktur der Arbeit

Nach dieser kurzen Einleitung werden im nachfolgenden Kapitel die Grundlagen für diese Arbeit erläutert. Weiter folgt eine Erläuterung darüber, wie Objekte zu betrachten, handhaben und verstehen sind. Jene werden als Kernelement der gesamten Problematik angesehen. Zusätzlich wird auch der Umgang mit den Objekten behandelt.

Danach beschäftigt sich Kapitel 4 mit der Konzeption eines Serversystems für die bereits beschriebene Problematik. Zunächst wird der Gesamtentwurf betrachtet und eine sinnvolle Unterteilung in einzelne Komponenten vorgenommen. Auf die Arbeitsweise und Kommunikation der einzelnen Komponenten, sowie dem Gesamtsystem, wird ebenfalls eingegangen.

In Kapitel 5 wird eine Beispielanwendung beschrieben und eine Realisierung des konzipierten Serverkonzeptes erarbeitet. Um den Rahmen dieser Bachelorarbeit nicht gänzlich zu sprengen, erfolgt die Realisierung aber nur unter Minimalkriterien, um die Funktionsfähigkeit des Konzeptes zu zeigen und eine Abschätzung über die Praxistauglichkeit zu ermöglichen.

Als Abschluss der Arbeit wird noch ein Fazit gezogen, um die gewonnen Erkenntnisse noch einmal aufzuzeigen.

2 Grundlagen

In diesem Kapitel werden die Grundlagen für diese Arbeit benannt und kurz beschrieben. Es werden vor allem im Vorfeld getroffene technische Vorgaben genannt, welche als Eckdaten fungieren und in den zu dieser Arbeit angefertigten Prototyp einer Beispielanwendung einfließen.

2.1 Begriff Spielwelt

Der Begriff Spielwelt bezieht sich auf die virtuelle Umgebung, in der sich der Spieler in einem Computerspiel befindet. Sie ist ein erdachtes und meistens auch rein fiktives Szenario. Die Spielwelt selbst umfasst die gesamte virtuelle Welt eines Spiels und auch alles, was in ihr zu finden oder anzutreffen ist.

2.2 Datenbank

Bei MMOGs fallen sehr viele Daten an. Im hier behandelten Szenario, einer Spielwelt bestehend aus Objekten, die alle als dynamisch, d.h. mit veränderlichen Attributen, anzusehen sind, ist das Datenaufkommen besonders hoch. Die Informationen zu allen in einer Spielwelt vorkommenden Objekten stellen sicherlich den Großteil des Datenaufkommens dar. Zusätzlich müssen noch benutzerbezogene Informationen hinterlegt werden. Dazu zählen auch sensible Daten wie Kontaktinformationen, Namen, Passwörter und für nicht kostenlos nutzbare Anwendungsfälle auch Kontodaten. Vor allem zu sensiblen Daten ist der Zugriff einzuschränken und Sicherheitsmechanismen gegen Missbrauch sind einzubauen.

Die Verwendung einer Datenbank zur Verwaltung und Speicherung dieser Daten ist unabdingbar. Da die dafür verwendete Software jedoch in jedem Anwendungsfall variieren kann, wurde im Vorfeld der Einsatz eines MySQL Servers der Version 5.5 für die Beispielanwendung festgelegt. Die Entscheidung für den Einsatz von MySQL beruht auf der vom Verfasser bevorzugten Verwendung von Open Source Datenbank Software und Vorerfahrungen im Umgang mit MySQL.

MySQL[4] ist ein Open Source Projekt der ehemaligen Firma Sun Microsystems und nun Teil von Oracle, einem großen Anbieter in Sachen Datenbanken. Die Vorteile sind unter anderem die Tatsache, dass MySQL auf sehr vielen Plattformen verfügbar ist, sowie Interfaces für alle gängigen Programmiersprachen angeboten werden.

[4] vgl.: Verbreitung von Datenbanken: <http://www.mysql.de/why-mysql/marketshare/>, verfügbar am 11.08.2011

Die Sicherung einer MySQL Datenbank ist mit sehr wenig Aufwand zu bewältigen. Da sie in einer Datei gespeichert wird, ist es problemlos möglich, jene zu kopieren und zu archivieren. Ein Wiederherstellen im Problemfall stellt daher auch kein Problem dar.

Eine separate Administration der Datenbank kann über die mitgelieferte Software erfolgen oder über verfügbare Web-Interfaces.

2.3 Netzwerkschnittstelle

Um den Entwicklungsaufwand zu verringern, liegt die Verwendung einer bestehenden Lösung und eine Schnittstelle zum Netzwerk nahe. Im Vorfeld der Anfertigung dieser Bachelorarbeit wurde bereits mit zwei verschiedenen Game-Engines gearbeitet. Eine davon, die Unity 3D Engine[5], verfügte über eine eingebaute Netzwerkengine, welche das erstellen von MMOG Anwendungen ermöglichte. Die Netzwerkengine der Firma Jenkins Software ist separat erhältlich.

Als Netzwerkschnittstelle kommt die RakNet API von Jenkins Software LLC zum Einsatz. Das Framework wird auf der Internetseite als 30-Tage-Testversion kostenlos zum Download angeboten.

Das von Jenkins Software als Netzwerk-Engine bezeichnete RakNet, Version 4.0, bietet alle im Bereich eines MMOGs erdenklichen Funktionalitäten[6] und ist auf minimale Ressourcennutzung hin optimiert. Als Funktionalitäten zu nennen sind unter Anderem Objekt Replikation, ein Lobby System, Verbindungsverschlüsselung, RPC, Autopatcher und Cloud Computing.

Unterstützt werden die Programmiersprachen C++ und C#.

RakNet findet Verwendung bei MMOGs von verschiedensten Anwendern. Laut Jenkins Software[7] wird es unter Anderem von Sony Online Entertainment[8], BIGPOINT[9], Gazillion[10] und vielen Weiteren verwendet.

[5] vgl.: Unity 3D Homepage: <http://unity3d.com/>, verfügbar am 11.08.2011
[6] vgl.: RakNet Featureliste: <http://www.jenkinssoftware.com/features.html>, verfügbar am 11.08.2011
[7] vgl.: Homepage von Jenkins Software: <http://www.jenkinssoftware.com/customers.html>, verfügbar am 10.08.2011
[8] vgl.: Homepage von Sony Online Entertainment: <http://www.soe.com/sonyonline/>, verfügbar am 11.08.2011
[9] vgl.: Homepage von BIGPOINT: <http://www.bigpoint.net/?lang=de>, verfügbar am 11.08.2011
[10] vgl.: Homepage von Gazillion Entertainment: <http://www.gazillion.com/\#/home>, verfügbar an 11.08.2011

2.4 Entwicklungsumgebung

Die Entwicklung der Beispielanwendung erfolgt unter Microsoft Windows. Da für den Prototypen keine Plattformunabhängigkeit angestrebt wird, fällt die Wahl der Entwicklungsumgebung auf das ebenfalls von Microsoft angebotene Visual Studio, Version 2010 Ultimate. Die Entwicklungsumgebung kann von Studenten über MSDNAA kostenlos bezogen werden. RakNet, die zur Anwendung kommende Netzwerkschnittstelle, wurde in C++ geschrieben und um ohne zusätzlichen Aufwand arbeiten zu können, wird die Beispielanwendung auch in C++ geschrieben. Für die Datenbank wird ebenfalls eine passende API angeboten. Dokumentationen für die Verwendung von RakNet und der Datenbankschnittstelle mit der Entwicklungsumgebung von Microsoft sind ebenfalls vorhanden. Dies rechtfertigt die getroffene Entscheidung für die Entwicklungsumgebung im Hinblick auf die Erstellung einer Beispielanwendung mit minimalem Aufwand, um den Rahmen einer Bachelorarbeit einhalten zu können.

Alternativen, wie das Eclipse Framework[11] wären auch möglich, wiesen aber teils erhebliche Nachteile im Hinblick auf die Dokumentation auf.

2.5 Art und grundlegende Funktionsweise des Servers

Bei Onlinespielen gibt es eine Vielzahl unterschiedlicher Modelle für die Organisation und Verwaltung der einzelnen Teilnehmer. Da sich diese Arbeit im Segment der MMOGs befindet, kommen nur Client-Server Modelle mit einem autoritären Server in Frage. Bei autoritären Servern liegt die Berechnung der Spielphysik, Bewegung der Spieler und weiterer Aktionen gänzlich auf Seiten des Servers. Der Client realisiert nur die Interaktion mit dem Benutzer und führt für sich Vorausberechnungen des zu erwartenden Spielverlaufs durch, welche dann mit dem Server abgeglichen werden. Dieses Modell ist auch am sichersten in Bezug auf ungewollte Spielmanipulation, da die Einflussmöglichkeiten des Nutzers klar eingeschränkt bleiben. Ein direkter Zugriff auf die Spielmechanik bleibt verwehrt.

Übliche Online-Spielkomponenten, wie Chat, Voice-Chat oder spielinterner Handel sind für die Entwicklung des Serverkonzeptes nicht zu berücksichtigen. Chat beziehungsweise Voice-Chat kann mit separaten Komponenten realisiert werden und hat keine Bedeutung für die Verwaltung, Bearbeitung und Berechnung der Spielwelt. Handelsfunktionalität kann bei Bedarf über Objektfunktionalität abgebildet werden.

[11] vgl.: Eclipse Homepage: <http://www.eclipse.org/, verfügbar am 10.08.2011

3 Objekte und Verteilung

In diesem Kapitel soll darauf eingegangen werden, wie der Begriff des Objektes im Umfeld dieser Arbeit zu betrachten ist. Des Weiteren wird auch auf die Verwendung von Objekten eingegangen, sowie deren Abbildung, technische Realisierung und Umsetzung.

3.1 Objektbegriff

Als Objekt wird alles bezeichnet, was in einer erdachten Spielwelt vorkommt. In einer Spielwelt existieren üblicherweise eine Vielzahl von Gegenständen und andersartigen Objekten. Nicht wenige davon können als “tot“ bezeichnet werden, da sie effektiv nur eine einzige Funktionen haben. Präsenz, ohne jegliche andere Funktionalität, wird genutzt um dem Spieler die Tatsache der “Leere“ einer Spielwelt vorzuenthalten. Somit wird praktisch realisiert, dass der Spieler sich nicht im leeren Raum aufhält, sondern in einer Spielwelt, voll von verschiedensten Objekten und Gegenständen.

Die Objekte, um welche es sich hier handelt, sind nur in Betracht auf den Server zu verstehen. Demnach sind nur Attribute und Eigenschafte zu berücksichtigen, welche für eine serverseitige Bearbeitung notwendig sind. Dinge wie Ton und Beleuchtung spielen hier keine Rolle. Physikalische Eigenschaften sind für die Berechnung am wichtigsten.

Im Hinblick auf die Tatsache, dass die meisten bisher verfügbaren Spielwelten rein oder überwiegend statisch sind, kann man sagen, dass eine Vielzahl der vorkommenden Objekte auch statisch sind. Sie haben keinerlei veränderliche oder beeinflussbare Attribute und bilden lediglich die Spielumgebung für den Nutzer ab. Der große Vorteil von statischen Objekten ist die Tatsache, dass sie nicht über das Netzwerk zum Client übertragen werden müssen. Zumindest nicht in Fällen, wo eine Client-Software Anwendung findet. Es existieren auch Verfahren, bei denen z.B. alle Daten zum Browser eines Clients mittels Streaming übertragen werden und dort eine Darstellung erfolgt. Das zuletzt genannte Verfahren wird hier nicht weiter Berücksichtigt. Statische Objekte sind in die Client-Software integriert und bei Änderungen an der Spielwelt wird die Client-Software, und damit auch statische Objekte, aktualisiert.

Bei dieser Arbeit werden alle Objekte als dynamisch angesehen, d.h. sie sind prinzipiell veränderbar und müssen somit auch regelmäßig an den Client übertragen werden. Jedes einzelne Element der Spielwelt verfügt somit über veränderbare Attribute und muss auch an entsprechender Stelle in der Datenbank abgelegt werden.

3.2 Unterteilung der Objekte in Gruppen und Klassen

In einer Spielwelt kommen die verschiedensten Objekte vor, alle haben Eigenschaften, Methoden und Attribute, welche auf alle zutreffen, eine Gruppe oder nur für sie allein. Als Grundlage sei die Position eines jeden Objektes im Spiel zu nennen. Unbedingt vorhanden sein sollten auch Informationen darüber, ob das Objekt Einfluss auf seine Umwelt nimmt oder nicht. Beispiele hierfür sind Lichtquellen oder andere physikalische Eigenschaften wie Masse, Gravitation und auch die Größe eines Objektes.

Um dieser Vielfalt gerecht zu werden, ist eine Unterteilung in Klassen und Gruppen unumgänglich. Eine genau Planung für die Gruppierung und erfolgt anhand des jeweiligen Anwendungsfalles. Es erleichtert auch die Datenhaltung. Jeder einzelne Objekttyp ist auf eine separate Klasse abzubilden. Verschiedene Objekttypen mit ähnlichen Eigenschaften und Attributen sind vorzugsweise von der gleichen Oberklasse abzuleiten.Praktisch zu realisieren ist dies über eine hierarchische Vererbungsstruktur. Die Basis oder auch Wurzel hierfür wird einfach als “GameObject“ bezeichnet. Für die Ablage und Speicherung in einer Datenbank verfügt jedes Objekt über eine einzigartige Kennung, ID, und die Information zu welcher Klasse es gehört. Anhand dessen können klassenspezifische Daten in der Datenbank effektiver verwaltet werden.

3.3 Verteilung von Objekten

Bei der Verteilung von Objekten kommen verschiedene Faktoren zum Tragen. Einer dieser Faktoren ist die Tatsache, dass eine möglichst hohe Parallelisierung erreicht werden soll, um technischen Einschränkungen aus dem Wege zu gehen. Dies bezieht sich hauptsächlich auf die Auslagerung auf verschiedene Hardware.

Eine Umsetzung dessen bietet der nicht neue Ansatz für eine Zuweisung von Gebieten. Jedes Objekt wird somit immer einer Zone, einem Gebiet bzw. einem Areal zugeordnet. Für alle Gebiete werden Grenzen definiert, entweder statisch oder dynamisch. Dynamische Grenzen erfordern mehr Verwaltungsaufwand, haben aber Vorteile im Hinblick auf eine Last- und Ressourcenverteilung. Als Zwischenlösung bietet es sich auch an, Gebiete bei Bedarf weiter zu unterteilen oder wieder zusammenzufassen. Auf diese Weise lässt sich ebenfalls Dynamik in einem statischen Umfeld abbilden.

Wenn sich ein Objekt - aufgrund von Bewegung - von einem Gebiet in eine anderes begibt, so ist eine Änderung in den zugehörigen Objekt-Attributen vorzunehmen. Da alle Gebiete als voneinander unabhängig zu betrachten sind, müssen für den Übergang Mechanismen definiert werden.

Jedes Objekt benötigt, nach der erfolgten Definition, Attribute für zonenspezifische Positionsdaten und weltbezogene, zonenneutrale Informationen.

4 Serverkonzept

In diesem Kapitel erfolgt die Konzeption eines Serversystems und eine Unterteilung in verschiedene Komponenten mit einer Beschreibung darüber, welche Aufgaben diese Übernehmen sollen.

4.1 Grundlegende Aufgabenverteilung

Einige Aufgaben, die es zu bewältigen gilt, wurden bereits in den vergangen Kapiteln genannt und andere nicht. Da diese jedoch noch nirgends komplett genannt und zusammengefasst wurden, wird dies in diesem Abschnitt erfolgen. Es handelt sich hierbei um eine gute Voraussetzung für den Grobentwurf, um alle Komponenten planen zu können.

Bekannte Aufgaben sind von vornherein die Kommunikation mit dem Client, welche noch genauer zu beschreiben ist. Der Zugriff auf die Datenbank, sowie interne und externe Kommunikation des gesamten System ist genauer zu definieren.

Ein Konzept für die Behandlung, Bearbeitung und Übertragung aller Objekte stellt den Kern des Systems dar.

Die Datenbank und Verbindung zu den Clients sind als externe Schnittstellen zu betrachten.

Für alle Komponenten und für die Verbindungsaufnahme der Clients muss das System eine einheitliche, bekannte und für alle erreichbare Adresse besitzen. Um dies zu realisieren bietet es sich an, eine Komponente mit reiner Vermittlungsaufgabe zu schaffen. Diese ist dann ausschließlich für die Vermittlung aller Komponenten verantwortlich. Es erfolgt keine Bearbeitung, nur eine Weiterleitung an jeweils zuständigen, andere Komponenten. Aus Sicherheitsgründen ist eine Trennung dieser Komponenten in interne und externe - Client bezogene - Kommunikation in Erwägung zu ziehen.

4.2 Datenbank-Kommunikation

Da der Kern des zu planenden Systems darin besteht, mit Objekten zu arbeiten, ist die Verwendung einer Datenbank notwendig. Die Datenbank selbst ist als externe Komponente zu betrachten. Somit liegt es nahe, die Aufgabe der Kommunikation mit der Datenbank einer eigenen Komponente zuzuweisen.

Das gesamte System arbeitet auf Grundlage der Netzwerkengine RakNet. Durch die

Zuweisung aller Interaktionen mit der Datenbank auf eine Komponente, wird nicht mehr als eine Schnittstelle für die externe Kommunikation an dieser Stelle benötigt. Für den Austausch der Datenbank fällt somit auch nur Arbeit an einer Stelle an und der Aufwand für eine solche Änderung wird minimiert. Da die verwendete Datenbanksoftware nicht festgelegt sein soll, ist die Verwendung einer zentralen Schnittstelle zur Datenbank mehr als nur angebracht.

Neben den Objektdaten müssen auch alle Nutzerdaten gespeichert und verwaltet werden. Im einfachsten Fall erfolgt dies in der gleichen Datenbank. Da aber aus verschiedenen, vor allem aber sicherheitsrelevanten, Gründen eine Trennung zwischen Nutzer- und sonstigen Daten erfolgen sollte, ist eine Unterteilung der Datenbankkomponente in Erwägung zu ziehen.

4.3 Client-Kommunikation

Die Kommunikation zwischen Client und Server lässt sich in mehrere Schritte und Aufgabenfelder unterteilen.

Zuerst muss der Client eine Verbindung zum Server aufbauen. Ist diese aufgebaut, so erfolgt der Login. Beim Login muss der Benutzer einen Name und ein Passwort eingeben, welche beide an den Server übermittelt werden. Der Server seinerseits muss nach Erhalt dieser Daten überprüfen, ob ein passender Datensatz in der Datenbank existiert. Der Client erhält entweder eine Erfolgs- oder eine Fehlermeldung. Im Fehlerfall kann der Benutzer erneut Name und Passwort eingeben, für den Fall, dass er sich geirrt oder vertippt hat. Die Anzahl der Login Versuche sind meistens begrenzt. Üblich sind drei bis fünf Falschversuche in einem 30 Minuten Intervall. Dieser Wert sollte frei konfigurierbar sein.

Die Aufgabe des Logins sollte von einer entsprechenden Komponente übernommen werden. Um die Nutzerdaten zu überprüfen braucht diese Komponente Zugriff auf die Datenbank mit den Benutzerdaten. Eine Verbindung zur Datenbankkomponente ist hierfür einzuplanen und zu realisieren. Nach der Anmeldung am System, dem Login, steht dem Spieler üblicherweise ein Auswahl zur Verfügung, über welche Objekte er die Kontrolle übernehmen möchte. In Rollenspielen ist dies meist eine Spielfigur, ein Charakter, in Rennspielen ein Fahrzeug, bei Flugsimulationen ein Flugzeug. Oft besteht auch die Möglichkeit z.B. neue Spielfiguren zu erstellen oder eigene zu verändern.

Das Erstellen und Bearbeiten von Spielfiguren und Ähnlichem benötigt Zugriff auf die Spieldatenbank. Eine Verbindung zur Datenbankkomponente sei hierfür einzuplanen und zu realisieren.

Alternativ könnte für die Wahl des zu steuernden Objektes auch ein anderes Modell Ver-

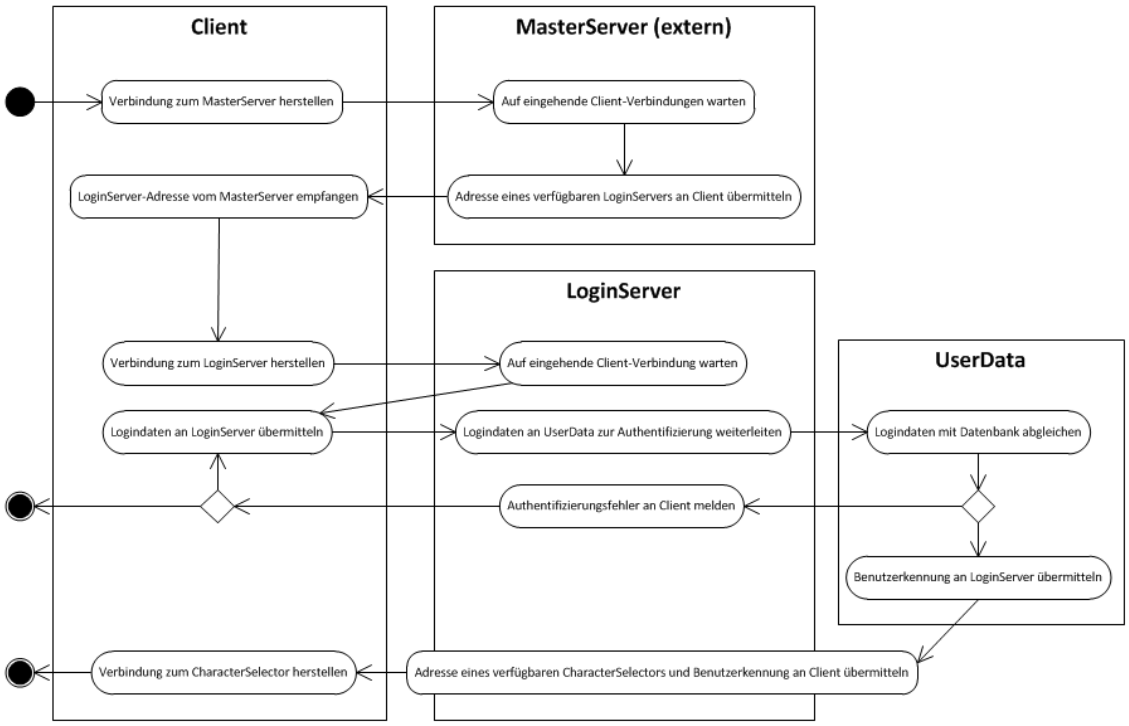

Abbildung 4.1: Client Login und Authentifizierung

wendung finden. Dem Spieler werden nur Objekte zur Wahl angeboten, welche bereits in der Spielwelt existieren, aber momentan von keinem Spieler gesteuert werden. Der Spieler selbst hat nicht die Möglichkeit sich im Vorfeld z.B. eine Spielfigur zu erstellen oder zu bearbeiten, er bekommt nur die Auswahl über vorhandene Elemente. Um eine solche Auswahl zu erhalten, ist es möglich, die Spielwelt nach verfügbaren Objekten abzusuchen oder aber eine Abfrage an die Datenbank zu senden, welche in dem Fall über die notwendigen Informationen verfügen muss. Die letzte Variante ist zu bevorzugen, da bei ihr nur ein Zugriff auf die Datenbank erfolgen muss und kein Aufwand für das restliche Serversystem anfällt.

Die gerade beschriebene Aufgabe verlangt ebenfalls nach einem Zugriff zur Spieldatenbank. Diese und die vorangegangene Aufgabe stehen sich sehr nahe, können und werden allerdings separat betrachtet.

Mit der Auswahl eines Objektes, über welches der Spieler die Kontrolle übernimmt, entsteht eine neue Aufgabe. Der Spieler muss mit Informationen über sein Objekt und das jeweilige Umfeld versorgt werden. Auf die Komponente, welche diese Aufgabe zuteil wird, entfällt eine besonders hohe Gewichtung. Es muss dem Spieler möglich sein, Befehle an sein Objekt zu senden, es zu steuern. Eine Aktualisierung der objektrelevanten Daten und der des Umfeldes ist in ausreichend kurzen Abständen zu gewährleisten.

Diese Komponente benötigt keine Verbindung zur Datenbank und eine weitere, genaue-

re Beschreibung wird später erfolgen. Sie übernimmt die Vermittlung und Weiterleitung von Objektdaten an die Clients.

4.4 Interne Objektbehandlung und Verteilung der zugehörigen Aufgabenbereiche

Auf die Objekte, deren Behandlung, Übertragung und Weiterleitung entfallen einige Aufgaben wofür verschiedene Komponenten geplant und realisiert werden müssen.

Vor Inbetriebnahme des Systems befinden sich alle Objekte und Daten in der Datenbank. Bevor das System gestartet werden kann, muss es erst initialisiert werden. Das bedeutet, alle Objekte müssen aus der Datenbank gelesen und an eine für die Berechnung zuständige Komponente übermittelt werden. Erst wenn alle Objekte ausgeliefert und bereit für eine Bearbeitung sind, kann der Start erfolgen.

Für die Aufgabe der Berechnung und Aktualisierung von Objektdaten ist eine Komponente zu planen und realisieren, die nur diese Tätigkeit ausführt. Auf sie werden alle Objekte aufgeteilt und sie führt auch die Berechnungen für Positions-, sowie Rotationsänderungen durch. Eine Datenbankverbindung ist für den Bezug der Objektdaten unerlässlich. Des Weiteren werden die aktualisierten Daten auch an die Datenbank weitergeleitet, um deren Änderungen darin zu speichern.

Um die Objektdaten auch an die Clients zu übermitteln, wird eine weitere Komponente konzipiert und realisiert. Sie erhält die aktualisierten Objektdaten von der zuvor beschrieben Komponente und sendet die jeweils relevanten Daten an die jeweiligen Clients weiter. Somit spielt diese Komponente eine reine Vermittlerrolle. Dies wurde auch bereits im voran gegangenen Abschnitt Client-Kommunikationërwähnt.

Somit werden die verschiedenen Aufgaben, die auf Objekte entfallen, auf drei Komponenten aufgeteilt. Zum einen die Datenbankkomponenten, danach jene, welche für die Berechnung und Bearbeitung der Objekte zuständig sind und eine Vermittlungskomponente für die Übertragung der Daten an die Clients.

4.5 Systeminterne Kommunikation zwischen den einzelnen Komponenten

Bisher wurden im Groben, mit Unterteilungen, acht Komponenten beschrieben, welche alle miteinander kommunizieren müssen. Dennoch müssen die meisten Komponenten nicht alle anderen kennen oder mit ihnen kommunizieren. Daher ist es von Vorteil, die Kommunikationswege festzuhalten.

Zentraler Punkt der Kommunikation ist die zu Beginn erwähnte Komponente, welche für alle bekannt und erreichbar sein muss. Im weiteren Verlauf des Dokumentes wird diese Komponente als MasterServer bezeichnet. Sie muss präsent sein, bevor alle anderen Komponenten eine Verbindung aufbauen können. Für die Unterteilung in eine Client- und eine Serverseite ist es notwendig, dass beide Seiten miteinander kommunizieren, denn der sich verbindende Client muss die Adresse einer verfügbaren Komponente für den Login Vorgang erhalten.

Die Komponente für den Login Vorgang wird folgend als LoginServer bezeichnet. Sie registriert sich wie alle anderen Komponenten nach ihrem Start beim MasterServer, was den ersten Kommunikationsweg dieser Komponente aufzeigt. Darüber hinaus muss sie auch Verbindungen von Clients annehmen, welche sich authentifizieren wollen. Um Clients authentifizieren zu können, müssen die empfangenen Daten noch mit den in der Datenbank verfügbaren abgeglichen werden. Die Verbindung zur Datenbank stellt den dritten Kommunikationsweg dar. Sobald der Client korrekt authentifiziert wurde, sollte er noch an die nächste Komponente weitergeleitet werden. Folgend als CharacterSelect bezeichnet, bietet sie dem Spieler eine Auswahl an Objekten, über welche er die Kontrolle über nehmen kann. Daraus ergibt sich der vierte Kommunikationsweg für den LoginServer.

Damit die Client-Authentifizierung erfolgen kann, muss die Datenbankkomponente für Benutzerdaten verfügbar sein. Sie wird folgend als UserData bezeichnet. Sie registriert sich ebenfalls zunächst beim MasterServer, welches den ersten Kommunikationsweg darstellt, und wartet auf Anfragen von der LoginServer-Komponente um Benutzerdaten abzugleichen. Dies ist der zweite Kommunikationsweg dieser Komponente.

Die CharacteSelect-Komponente verfügt über fünf Kommunikationswege, der erste ist die Verbindung zum MasterServer. Die Zweite stellt die Verbindung zur LoginServer-Komponente dar, die Dritte ist die Verbindung zur Komponente für die Objektdaten, folgend als WorldData bezeichnet. Als Viertes ist die Verbindung zu Clients ein weiterer Kommunikationsweg und darüber hinaus als fünftes die Verbindung zu der Objektkomponente, welche die Weiterleitung der Objektdaten an die Clients übernimmt, folgend als GOaID bezeichnet.

Die GOaID-Komponente meldet sich ebenfalls beim MasterServer an und hat einen Kommunikationsweg für Clients. Der dritte Kommunikationsweg ist die schon erwähnte Verbindung zur CharakterSelect-Komponente. Ein vierter Weg besteht mit der Verbindung zur Komponente für die Berechnung und Aktualisierung der Objektdaten, folgend als WorldProcessor bezeichnet.

Die WorldProcessor-Komponente verfügt über Verbindungen zu MasterServer und GOaID-Komponente. Für den Bezug der Objektdaten und zum Abspeichern selbiger kommuniziert sie mit der WorldData-Komponente.

Die Kommunikationswege der UserData-Komponente sind neben ihre Verbindung zum MasterServer bereits genannt. Der zweite der beiden Kommunikationswege ist jener zum LoginServer.

Für die WorldData-Komponente sind auch bereits alle Kommunikationswege genannt. Sie kommuniziert mit MasterServer-, CharacterControl- und WorldProcessing-Komponente.

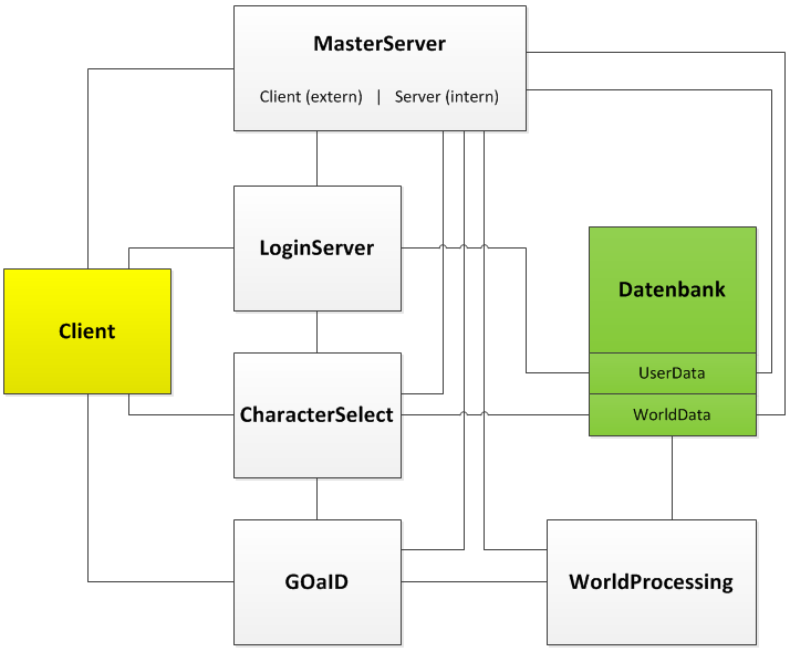

Abbildung 4.2: Kommunikationswege der einzelnen Systemkomponenten

4.6 Genauere Beschreibung der Server-Komponenten

In diesem Abschnitt wird nun etwas genauer auf die Funktionsweise der zuvor definierten System-Komponenten eingegangen.

Der Server soll möglichst beliebig viele Objekte dynamisch verwalten können und jedem Nutzer, die jeweils relevanten Informationen zukommen lassen. Um dies zu realisieren, muss die Möglichkeit bestehen, dem Server weitere Ressourcen hinzuzufügen. Eine Parallelisierung von Prozessen und Ressourcen soll ebenfalls berücksichtigt werden. Im Betrieb ist eine Umverteilung von Ressourcen zur Gewährleistung des reibungsfreien Ablaufs von Nöten.

4.6.1 MasterServer Komponente

Zentrale Komponente des Systems ist der MasterServer, bei dem sich zunächst alle anderen Serverkomponenten anmelden. Um die Last an dieser kritischen Stelle möglichst gering zu halten, übernimmt diese Komponente nur Verwaltungs- beziehungsweise Vermittlungsaufgaben. Von ihr ausgehend werden auch übergeordnete Vorgänge gestartet. Darunter ist das Initialisieren für eine Erstinbetriebnahme, sowie das Starten und Pausieren der Objektbearbeitung und Auslieferung. Der MasterServer verweist sich verbindende Clients auch an die nächste Instanz im System, dem LoginServer. Da aus Lastverteilungsgründen mehrere LoginServer-Komponenten vorhanden sein können, obliegt es dem MasterServer die Zuweisung vorzunehmen.

Um eine effektive Verteilung vornehmen zu können, führt der MasterServer Statistiken über alle registrierten Komponenten. Somit ist dem MasterServer auch bekannt, über welche Ressourcen das Gesamtsystem verfügt. Darüber hinaus senden alle Komponenten auch regelmäßig Informationen über ihre Auslastung. Auf diese Weise können Engpässe erkannt und gegebenenfalls eine Umverteilung veranlasst werden.

4.6.2 LoginServer Komponente

Die für die Clients zuständige Stelle wird als LoginServer bezeichnet. An ihn werden Clients vom MasterServer weiter verwiesen, um sich am System anmelden zu können. Nachdem sich ein Client zum LoginServer verbunden hat, erfolgt eine Authentifizierung. Um sich zu authentifizieren sendet der Client einen Benutzernamen und ein Passwort. Diese Daten werden mit den in der Datenbank vorhandenen abgeglichen. Um dies zu ermöglichen sendet der LoginServer seinerseits eine Anfragen an die dafür zuständige Datenbankkomponente, UserData. Wenn ein passender Datensatz in der Datenbank gefunden wurde, antwortet die UserData-Komponente mit der zugehören User-ID.

Es ist in Erwägung zu ziehen, die Verbindung zwischen Client und Serversystem ab dem Zeitpunkt des Verbindungsaufbaus zwischen Client und LoginServer zu verschlüsseln.

Für den Fall, dass die übermittelten Benutzerdaten nicht in der Datenbank gefunden wurden, gilt es dies dem Client mitzuteilen. Die Meldungen “Falsches Passwort“ oder “Benutzername unbekannt“ sind hier zweckmäßig. Dem Client wird ermöglicht, erneut Nutzerdaten zu senden. Eine Einschränkung der Fehlversuche ist aus Sicherheitsgründen zu empfehlen. Üblich sind drei bis fünf Fehlversuche in einem Intervall von 30 Minuten.

An dieser Stelle oder durch diese Komponente kann es dem Nutzer ermöglicht werden, eine neues Benutzerprofil anzulegen. Wenn diese über das Serversystem reali-

siert werden soll, ist es an dieser Stelle einzufügen. Alternativ kann dies auch über Webportal angeboten und realisiert werden. In dem Fall ist weder eine entsprechende Client-Software noch sonstig Aufwand notwendig. Die Pflege der Benutzerdaten kann somit auch über ein externes System, in direkter Verbindung zur Datenbank, geschehen.

Erhält der LoginServer von der UserData-Komponente die Meldung einer erfolgreichen Authentifizierung und eine User-ID oder auch eine Benutzerkennung, so wird der Client im System erneut weiter geleitet. Der Client wird an eine verfügbare CharacterSelect-Komponente verwiesen, ihr werden vom LoginServer die Verbindungskennung und User-ID übermittelt.

Die Verbindungsdaten für eine Weiterleitung an eine verfügbare CharacterSelect-Komponente bezieht der LoginServer wiederum vom MasterServer.

Da eine jede LoginServer-Komponene immer nur eine vorher definierte Anzahl an Clients bearbeiten kann, können je nach Notwendigkeit mehrere Instanzen dieser Komponente im Gesamtsystem existieren.

Prinzipiell muss immer mindestens eine Instanz dieser Komponente im System vorhanden sein. Für eine Umlagerung von Ressourcen wird beispielsweise an andere Stelle eine neue Instanz dieser Komponente in Betrieb genommen und die bisher präsente Instanz beendet. Das Beenden einer Instanz setzt natürlich voraus, dass keine Clients mehr mit ihr verbunden sind. Um dies zu gewährleisten sind Schutzmechanismen zu realisieren. Einerseits muss der Aufbau von neuen Clientverbindungen unterbunden werden und andererseits müssen alle noch verbundenen Clients abgearbeitet werden, bevor die Instanz sich beendet. Um die Zuweisung von neuen Clients zu unterbinden, wird beim MasterServer mitgeteilt, dass die Instanz des LoginServers sich beendet. Der MasterServer wird diese Instanz somit nicht mehr als verfügbar werten und somit auch für keine Zuweisungen mehr in Betracht ziehen.

4.6.3 CharacterSelect Komponente

Die CharacterSelect-Komponente bietet dem Spieler einen Einstiegspunkt in die Spielwelt. Sie bietet eine Auswahl an Objekten, über welche der Spieler die Kontrolle übernehmen kann. Um dem Benutzer diese Auswahl anbieten zu können, wird eine Anfrage mit der jeweiligen User-ID an die Spieldatenbank, WorldData-Komponente, gesendet.

Anhand der User-ID bekommt die CharacterSelect-Komponente eine entsprechende Auswahl, zur Verfügung stehender Objekte, von der Datenbank-Komponente geliefert. Zu den Objekten, aus welchen der Benutzer wählen kann, gehören auch Systeminformationen darüber, von welcher Verwaltungskomponente diese Objekte gerade verwaltet werden. Denn nach der Wahl eines Objektes, über welches Kontrolle erlangt

werden soll, muss der Client zur dafür zuständigen Verwaltungskomponente weiter verbunden werden. Die Aufgabe der Verbindungsweiterleitung muss ebenfalls von der CharacterSelect-Komponente übernommen werden. Die im System vorhandenen Statusinformationen finden hierfür Verwendung. Darüber hinaus sollten die Verwaltungsinformationen auch in der Datenbank hinterlegt werden. Entweder Datenbank oder MasterServer müssen Auskunft geben können.

Ist es im bestimmten Anwendungsfall vorgesehen, dass ein Spieler sich selbst Objekte erstellen kann, z.B. eine eigene Spielfigur, ein eigenes Fahrzeug oder ein anderes anwendungsspezifisches Objekt, so ist dies mit dieser Komponente zu realisieren. Entsprechende Funktionalitäten und Methoden sind zu implementieren.

Für den Erstellvorgang sind teils in der Datenbank hinterlegte Vorgaben und Bausteine zu benutzen oder alle Vorgaben sind seitens der Client-Software realisiert. In dem Fall müssen Kontrollmechanismen für das korrekte Erstellen von Objekten eingebaut werden. Generell sollte eine Prüfung auf Korrektheit eines Objektes am Ende eines Erstellvorgangs erfolgen, bevor das neue Objekt an die Datenbankkomponente übermittelt wird. Eine weitere Prüfung könnte durch die Datenbankkomponente erfolgen, in Hinsicht auf eine effektive Ressourcennutzung kann davon allerdings abgesehen werden um keinerlei zusätzlichen Arbeitsaufwand für die Datenbankkomponente zu erzeugen.

Hat der Nutzer ein Objekt gewählt, über das er die Kontrolle übernehmen will, so wird er von der CharacterSelect-Komponente an die für das Objekt zuständige Verwaltungskomponente, GOaID, weiter geleitet.

4.6.4 Datenbankkomponenten

Die Datenbankkomponenten realisieren alle Zugriffe auf die Datenbank. Sie sollten nur Anfragen auf die Datenbank ausführen. Dies ist bei der Konzeption und Realisierung auch zu beachten um an dieser Stelle unnötige Last zu vermeiden. Es werden zwei Komponenten für den Zugriff auf die Datenbank konzipiert. Dies ermöglicht eine Trennung von Nutzer- und System- bzw. Objektdaten.

4.6.4.1 UserData Komponente

Die UserData Komponente realisiert eine Verbindung zu den Benutzerdaten. Da diese Komponente den Datenbankzugriff zu allen anderen Daten abgrenzt, kann bei Bedarf im jeweiligen Anwendungsfall auch eine andere Datenbank benutzt werden.

Weitere Aufgaben entfallen nicht auf diese Komponente. Demzufolge meldet sie sich nur nach ihrer Initialisierung beim MasterServer an und wartet auf LoginServer, die sich mit ihr verbinden und Benutzer-Authentifizierungsanfragen stellen. Empfangene Nut-

zerdaten werden mit der Datenbank abgeglichen und im Positivfall wird eine hinterlegte User-ID, Kennung, an den LoginServer gesendet. Im Negativfall wird der LoginServer nur Informiert, dass die Benutzerdaten nicht korrekt sind. Eine Auskunft darüber, ob der Benutzername unbekannt oder das Passwort falsch war, kann den Informationen hinzugefügt werden. Auf eine solche Unterscheidung wird aus sicherheitsrelevanten Gründen im Anwendungsfall ebenso verzichtet. Diese Informationen können aber dennoch zumindest bis zur LoginServer-Komponente weitergeleitet werden, welche sie dann für statistische Zwecke zumindest temporär speichert.

4.6.4.2 WorldData Komponente

Die WorldData Komponente ist zentrale Schnittstelle für den gesamten Datenverkehr, der sich auf die Spielwelt bezieht. Zum Einen wird die CharacterSelect-Komponente mit allem benutzerspezifischen Informationen für einen Beitritt zur Spielwelt versorgt und zum Anderen erfolgt die Verteilung alle Objekte der Spielwelt an die dafür zuständigen WorldProcessing Komponenten.

Bei der Initialisierung des Gesamtsystems, der Vorbereitung für den Start, übernimmt diese Komponente auch wichtige Aufgaben. Einerseits werden alle Objektbereichsunterteilungen aus der Datenbank ermittelt und auf alle Bearbeitungskomponenten (WorldProcessing) verteilt. Andererseits werden auch alle den jeweiligen Zonen unterstehenden Objekte aus der Datenbank gelesen und an die dafür verantwortlichen Komponenten übertragen.

Solange das Gesamtsystem am Laufen ist, werden ständig aktualisierte Daten der Objekte an die WorldData Komponente gesendet. Diese gilt es auch in der Datenbank zu aktualisieren.

Von der CharacterSelect Komponente neu erstelle Objekte müssen natürlich auch in der Datenbank abgelegt und an eine zuständige WorldProcessing Komponente ausgeliefert werden. Entsprechende Mechanismen hierfür sind zu realisieren.

4.6.5 WorldProcessing Komponente

Die WorldProcessing Komponente übernimmt alle Aufgaben bezüglich der Bearbeitung und Berechnung von Objekten.

Nach dem Start der Komponente verbindet sie sich zunächst zum MasterServer, um dann einer GOaID Komponente unterstellt zu werden. Die Funktionsweise und Aufgabenbereiche werden später behandelt.

Nachdem die WorldProcessing Komponente einem GOaID zugewiesen und die Verbin-

dung hergestellt wurde, wartet sie auf die Zuweisung mindestens einer Zone von Objekten, für welche sie die Bearbeitung durchführen soll. Es können beliebig viele Zonen zugewiesen werden.

Wie schon im Kapitel “Objekte“ beschrieben, sind alle Objekte der Spielwelt Zonen zugewiesen, welche selbst noch eine feinere Unterteilung erlauben. Wenn auf eine Zone so viele Objekte entfallen, dass eine WorldProcessing Komponente allein nicht mehr in der Lage ist, sie in ausreichend kurzer Zeit zu bearbeiten, besteht die Möglichkeit die Zone weiter zu unterteilen oder ihre Grenzen zu verlagern. Mit einer Verschiebung der Zonengrenzen kann bewirkt werden, dass ein Teil der Objekte in eine andere Zone über geht, auch ohne, dass die Objekte sich bewegt hätten.

Sobald die Zuweisung einer Zone erfolgt ist, sendet die WorldProcessing Komponente eine Anfrage auf Objektzuweisung an die WorldData Komponente. Diese Anfrage quittiert der WorldData Komponente auch den Erhalt der Zonenzuweisung und signalisiert Empfangsbereitschaft für Objektdaten.

Einer einzelnen WorldProcessing Komponente können natürlich auch mehrere Zonen zugewiesen werden. Die Bearbeitung der einzelnen Zonen erfolgt jedoch voneinander unabhängig. Die verschiedenen Bearbeitungseinheiten, eine für jede Zone, arbeiten parallel. Praktisch kann dies mit je einem Thread pro Zone realisiert werden, insofern keine weitere technische Unterteilung angestrebt wird.

4.6.6 GameOrganisationAndInformationDistribution Komponente (GOaID)

Die GOaID Komponente übernimmt vorwiegend Verwaltungstätigkeiten. Sie hat die wichtige Aufgabe die Informationen der Spielwelt, welche von den WorldProcessing Komponenten immer aktuell gehalten werden, an die Spieler weiter zu leiten. Wichtig ist dabei vor allem, dass jeder Spieler nur die für ihn relevanten Daten gesendet bekommt. Anders herum betrachtet muss der Client auch alle für ihn wichtigen Informationen bekommen. Daher sollten auch genaue Abgrenzungen definiert werden.

Eine jede GOaID Komponente erhält immer nur Informationen von den ihr unterstellten WorldProcessing Komponenten und somit den Zonen, die von ihnen verwaltet werden. Aus diesem Grund sollte im System auch genau bekannt sein, welche GOaID Komponenten welche Zonen verwalten, denn die CharacterSelect Komponenten müssen die Clients auch an die entsprechende Stelle weiterleiten können. Um dies zu gewährleisten sollten Statistiken geführt werden, die zumindest dem MasterServer oder auch der WorldData Komponente bekannt sind. Die CharacterSelect Komponente muss somit nur bei einer der zuständigen Auskunftsstellen anfragen, wohin der Client für den Eintritt in die Spielwelt weitergeleitet werden muss. Diese Auskunftsinformationen müssen

im ganzen System aktuell gehalten und repliziert werden. Dazu muss ein jeder GOaID lediglich bei Änderungen darüber informieren, wie seine aktuelle Zuständigkeit aussieht. Ein fester Intervall, in dem pauschal eine Aktualisierung auch ohne vorausgegangene Änderung erfolgt, ist auch denkbar, wenn auch nicht von Nöten.

5 Beispielanwendung

In diesem Kapitel wird auf die im Rahmen dieser Arbeit realisierte Beispielanwendung eingegangen.

Für die Erstellung einer Beispielanwendung muss zunächst ein Anwendungsfall definiert und zumindest grob beschrieben werden, bevor an eine Umsetzung des konzipierten Serverkonzeptes zu denken ist.

5.1 Festlegung auf einen Anwendungsfall

Zuerst gilt es einen bestimmten Anwendungsfall festzulegen. Onlinespiele kann man in verschiedene Gruppen unterteilen. Darunter zählen unter Anderen Rollenspiele, Strategiespiele, Simulationen und sogenannte First-Person-Shooter Spiele, welche im MMOG Segment eher weniger anzutreffen sind. Die Gruppe der First-Person-Shooter stellt sehr hohe Anforderungen an die Verbindung zum Server, da schnelle Bewegungen der Spielfiguren viel Datenaustausch mit dem Server bedeutet. Bei schnellen Bewegungen von Objekten im Spiel muss auch eine sehr hohe Genauigkeit für die Berechnung realisiert werden. Dies erfordert möglichst kurze Intervalle für die Berechnung. Bei der Gruppe der Rollenspiele, bei deinen ein Spieler eine Spielfigur innerhalb einer virtuellen Welt, der Spielwelt, steuert, können ähnlich hohe Anforderungen gestellt werden.

Für die Realisierung einer Spielwelt, welche komplett aus dynamischen Objekten bestehen soll und worauf das Konzept auch abzielt, eignet sich ein Weltraumszenario bestens. Von Grund auf ist der Weltraum leer und jedes Objekt, welches sich darin befindet, kann als Objekt in der Spielwelt abgebildet werden. Vom Spieler zu kontrollierende Objekte können Raumschiffe oder einfache Sonden sein. Anders ausgedrückt kann der Spieler die Kontrolle über Objekte mit einem Antrieb übernehmen und somit deren Verhalten, wenn auch nur die Bewegung, beeinflussen.

Neben den für Spieler kontrollierbaren Objekten können noch zahlreiche andere Objekte in der Spielwelt abgebildet werden. Verschiedenste Arten von Himmelskörpern und physikalische Phänomene kommen hierfür in Frage.

Dieses Szenario beinhaltet optimale Voraussetzungen für die Realisierung einer minimalen Beispielwendung. Die Arten der Objekte für die Spielwelt lassen sich durch die gegeben Eigenschaften bestens abgrenzen.

5.2 Gestaltung der Spielwelt

Als Basis dient der dreidimensionale Raum, beziehungsweise Weltraum. Der Raum selbst ist von Grund auf als leer zu betrachten. Alles, was sich darin befindet, ist als Objekt zu betrachten und wird auch technisch so umgesetzt. Da eine Vielzahl von verschiedenen Objekten auftreten kann und wird, die jedoch alle gemeinsame Grundeigenschaften teilen, bietet es sich an, diese als Klassenhierarchie umzusetzen. Jedes Objekt verfügt somit über allgemeine Grundeigenschaften und objektspezifische Eigenschaften. Über einheitliche Funktionsaufrufe kann das objekttypische Verhalten berechnet und abgerufen werden.

Grundtypen für Objekte im Raum sind beispielsweise Sterne, Planeten, Monde, Kometen, Asteroiden und, um dem Spieler auch etwas zu bieten, Raumschiffe. Basisattribute für diese Objekte sind unter anderem Masse, Position im Raum, Rotation, Bewegung und Größe beziehungsweise Ausdehnung, damit verbundene Kollisionsbereiche und erforderliche Abfragen.

Objekte mit denen der Benutzer interagieren kann sind vorwiegend Raumschiffe. Sterne, Planeten, Monde, Kometen, Asteroiden und weitere sind für den Spieler hauptsächlich Hindernisse, da mit ihnen kaum Interaktion möglich ist, zumindest keine für den Spieler positive Interaktion.

5.3 Spielsteuerung, Einflussmöglichkeiten der Nutzer

Um die Komplexität der Benutzerschnittstelle in Grenzen zu halten, wird dem Spieler nur die Möglichkeit der Remotesteuerung von Raumschiffen gewährt. Er bekommt von der LoginServer Komponente eine Auswahl an verfügbaren Schiffen gestellt, aus der er sich die Kontrolle über jeweils ein Raumschiff zuweisen lassen kann.

Sobald der Spieler die Kontrolle über ein Raumschiff erlangt hat, kann er verschiedene Befehle erteilen. Die Steuerung der Schiffe wird bewusst indirekt gehalten, um Schwachstellen der Technik, beispielsweise die Verzögerung der Übertragung zum Server und zurück, zu kompensieren. Dies bezieht sich hauptsächlich auf Signallaufzeiten und Rechen- sowie Verwaltungsaufwand auf Serverseite. Mit der Vermeidung von abrupten Bewegungen entsteht eine höhere Toleranz für Signallaufzeiten und ein geringerer Synchronisationsaufwand. Befehle, die der Spieler geben kann, sind zum Beispiel der Kurs des Raumschiffs oder die Angabe eines Flugziels[12].

Raumschiffe, welche von keinem Spieler gesteuert werden, bekommen Aufgaben von einer simplen KI zugewiesen. Der Spieler kann einem Raumschiff auch Aufgaben zu-

[12] vgl.: Existierende Spiele mit derartigen Szenarien sind “Eve-Online“ von CCP oder die “X“-Serie von Egosoft

weisen, die weiterhin ausgeführt werden, wenn die Kontrolle wieder freigegeben wurde. Schiffe ohne Befehle stehen und treiben durch den Raum, sie verhalten sich in dem Fall weitestgehend wie Himmelkörper. Das Hauptunterscheidungsmerkmal ist die Benutzung eines Antriebs.

5.4 Technische Details der Beispielanwendung

Um Implementierungsaufwand für die Beispielanwendung zu verringern wurde auf eine Teilung der MasterServer Komponente zwischen Client und Systemseite verzichtet. Die Aufgaben der CharacterSelect Komponente werden vom LoginServer übernommen. Zusammengefasst wurden auch die beiden Datenbankkomponenten. WorldData übernimmt die Funktionalitäten von UserData.

Eine Dummy Konsolenanwendung bildet Client-Funktionalitäten ab. Sie wurde zusammen mit dem Serversystem entwickelt und verfügt über die Fähigkeit zur Verbindungsaufnahme mit dem Serversystem. Sie kann sich mit den über die Konsole eingegebenen Benutzerdaten authentifizieren und ermöglicht minimale Benutzerinteraktion mit der Spielwelt. Statusinformationen können zudem ausgegeben werden.

Alle Komponenten des Serversystems wurden als Konsolenanwendungen realisiert.

5.4.1 Details zur MasterServer Komponente

Da der MasterServer der Aufgabe nachkommt, für alle anderen Komponenten erreichbar zu sein und über Informationen für den Aufbau der Verbindungen zwischen den anderen Komponenten untereinander verfügt, wartet er auf sich verbindende Systemkomponenten oder Clients.

Sobald sich eine Systemkomponente mit dem MasterServer verbindet, werden die Verbindungsinformationen zentral gespeichert. Da alle anderen Systemkomponenten nach ihrer Initialisierung zunächst über keinerlei Informationen über andere Komponenten verfügen, übermittelt der MasterServer nach erfolgreicher Herstellung einer Verbindung die für die neu verbundene Komponente notwendigen Systeminformationen. Diese umfassen hauptsächlich Verbindungsdaten zu anderen, bereits verbundenen, Systemkomponenten.

Die Bearbeitung für jeden Komponententyp erfolgt voneinander unabhängig. Um eine Parallelisierung der einzelnen Aufgaben zu realisieren, erfolgt die Abarbeitung in Threads. Für jeden Komponententyp wird ein Thread angelegt. Jeder dieser Threads realisiert einen Kommunikationskanal.

5.4.2 Details zur WorldData Komponente

Zentrale Datenbankschnittstelle ist die WorldData Komponente. Sie kommuniziert mit einem MySQL Server Version 5.5 von Oracle. Um mit dem Server auch kommunizieren zu können, findet der MySQL Connector C++ Version 1.1.0 Verwendung. Der Connector umfasst einige Header-Dateien, eine DLL und dazugehöre Bibliotheken.

Um die Verflechtungen mit der Datenbank möglichst gering zu halten, wurden alle Funktionen mit Bezug auf die Datenbank in eine separate Header-Datei ausgelagert. Änderungen an der Datenbanksoftware verursachen somit nur Anpassungen an einer einzigen Datei.

Die Benutzung des MySQL C++ Connectors erfordert zusätzliche die Verwendung der Boost Bibliotheken. Diese sind Open Source Software.

5.5 Initialisieren und Starten des Systems

Für den Start, beziehungsweise das Initialisieren des Kernsystem, ist zunächst je eine WorldData, WorldProcessing, GOaID und MasterServer Komponente notwendig. Sind jene präsent, so kann vom MasterServer aus die Initialisierung veranlasst werden. Um den Befehl zu erteilen, wird “init“ ins Konsolenfenster der MasterServer Komponente eingegeben und bestätigt.

Sobald der Befehl zur Initialisierung gegeben wurde, wird dieser zu GOaID und WorldData weitergeleitet. Daraufhin ermittelt die WorldData Komponente alle in der Datenbank enthaltenen AreaTags, d.h. alle Gebiete, Zonen (Grobunterteilung) und alle Spielobjekte, die ihnen unterstehen. Anhand der Anzahl der AreaTags und zugehöriger Objekte wird eine Verteilung auf die verfügbaren WorldProcessing Komponenten veranlasst. Jeder dieser Komponenten wird mindestens ein AreaTag zugewiesen. Sind weniger AreaTags als WPs vorhanden, so ist eine weitere Unterteilung der festgelegten Gebiete vorgesehen, sodass es nicht weniger AreaTags als verfügbare WorldProcessing Komponenten gibt. Sind es mehr AreaTags als WPs, so werden den verfügbaren WPs weitere AreaTags anhand des Durchschnitts der Objekte pro Zone zugewiesen. Auf diese Weise soll eine möglichst gleichmäßige Grundverteilung der Objekte erreicht werden.

Mit Abschluss der Aufteilung der AreaTags, wird diese Zuweisung von der WorldData Komponente an alle WPs übermittelt. Mit Erhalt der AreaTags beginnen die WorldProcessing Komponenten ihrerseits mit der Initialisierung. Für jeden zugewiesenen AreaTag erfolgt eine separate Bearbeitung. Die Bearbeitungsroutine für jeden AreaTag wird mit dessen Zuweisung, auf dem WP erzeugt. Bearbeitungsroutinen werden mit der ID des AreaTags erzeugt und senden daraufhin eine Anfrage auf Objektzuweisung an die

WorldData Komponente mit dem dazugehörigen AreaTag. Die WorldData Komponente bezieht mit Erhalt der Anfrage auf Objektzuweisung nun alle benötigten objektspezifischen Daten für den AreaTag aus der Datenbank und übermittelt sie dem zugehörigen WP.

Sobald die Objektdaten beim WP eingetroffen sind und an die Bearbeitungsroutine weitergegeben wurden, beginnt die Befüllung der Objektbearbeitungsschleife mit den bezogenen Daten. Die empfangenen Objektdaten werden der WorldData Komponente noch quittiert. Diese sammelt alle Meldungen und erstellt eine Statistik darüber, wie viele Daten versendet und empfangen wurden.

Nach Auslieferung und Quittierung aller Objektdaten ist die Initialisierung komplettiert und die WorldData Komponente benachrichtigt daraufhin den MasterServer. Die Meldung über den Abschluss der Initialisierung signalisiert dem MasterServer, dass das System fertig zum Start ist. Darüber hinaus sendet er eine Nachricht über die Bereitschaft des Systems zum Start an alle verbundenen Komponenten.

Nun kann der Startbefehl erteilt werden. Für die Erteilung des Befehls wird “start“ in die Konsole des MasterServers eingegeben und bestätigt. Der Startbefehl veranlasst bei allen WPs, dass nun die Update-Methode für alle Objekte in jedem Bearbeitungsintervall aufgerufen und ausgeführt wird. Diese Methode beinhaltet alles für die Berechnung der Objekte notwendige. Im Pause-Modus und nach der Initialisierung bis zum Start findet keine Berechnung statt. Das System befindet sich zu der Zeit sozusagen im Leerlauf.

6 Fazit

Im Rahmen dieser Arbeit wurde ein Serverkonzept für die Verwaltung und Bearbeitung von dynamischen Objektdaten für Onlinespiele im MMOG Segment entwickelt. Das Konzept umfasst acht Systemkomponenten, die alle näher beschrieben wurden. Um eine möglichst hohe Skalierbarkeit in Bezug auf Hardware und Ressourcen zu gewährleisten, lassen sich alle Komponenten, mit Ausnahme der globalen Verwaltungseinheit und den Datenbankschnittstellen, in beliebiger Anzahl im System einsetzen.

Für das Konzept wurde ein Anwendungsfall in den Grundzügen beschrieben und eine Beispielimplementierung realisiert. Diese verfügt über minimale Funktionalität und reicht aus, um das Gesamtsystem in Betrieb zu nehmen.

Die Durchführung eines Praxistests, um herauszufinden wie viele Clients mit unterschiedlichen Hardwarekontingenten bedient werden können, war mangels entsprechender Hardware und leider möglich. Dies kann und sollte im Anschluss an diese Arbeit allerdings noch erfolgen.

Mit dem Sammeln von Informationen über das Verhalten der einzelnen Systembestandteile im Einsatz mit vielen Clients, sollte die Konzeption auf ihr Tauglichkeit hin untersucht werden. Die Verfahrensweise des mehrmaligen Weiterleitens der Clients von einer Systemkomponente zu einer andere, sei auch zu Prüfen. Bei nicht zufriedenstellenden Ergebnissen sollte eine Überarbeitung des Konzept dahingehend erfolgen, ob nicht eine weitere Komponente zwischen dem bestehenden System und den Clients eingeführt werden kann. Auch im Hinblick auf die Sicherheit der nach außen erreichbaren und damit angreifbaren Systemkomponenten, sollte die Einsatzfähigkeit nachgewiesen werden.

Anhang A: Konzeptionsskizzen

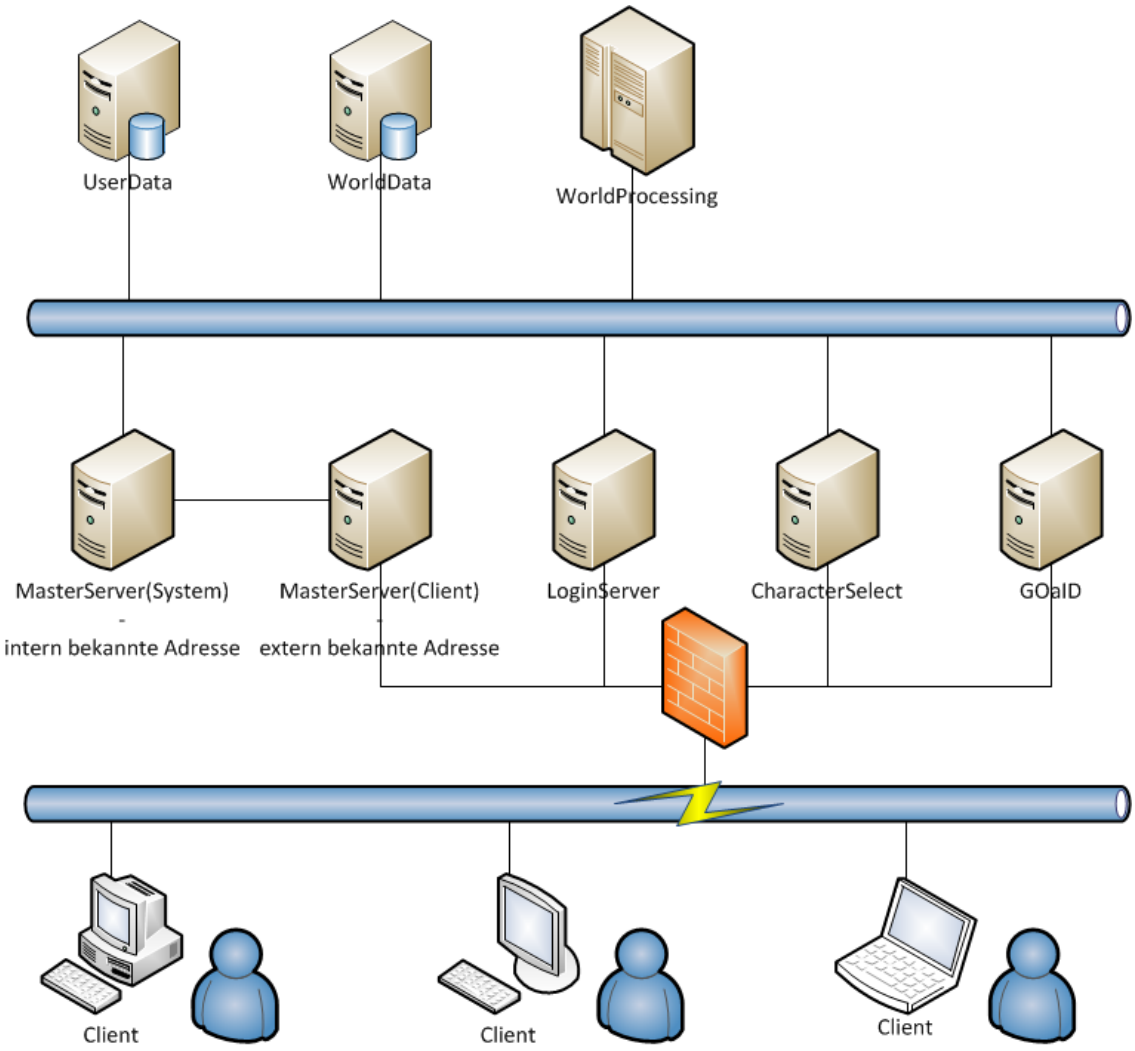

Abbildung A.1: Strukturplan des Serversystems

Anhang B: Screenshots der Beispielanwendung

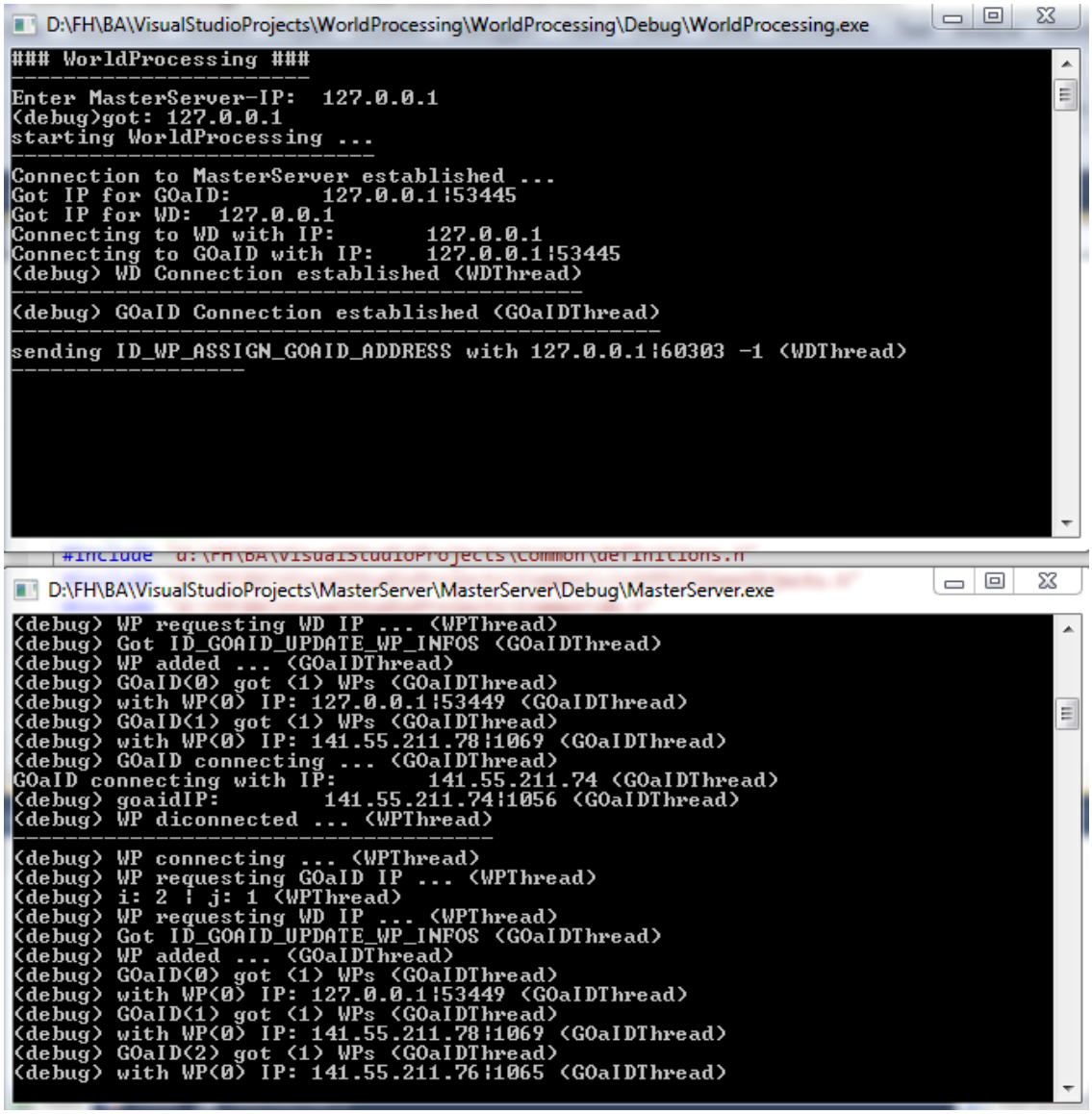

Abbildung B.1: Verbindungsaufbau der Komponenten MasterServer und WorldProcessing

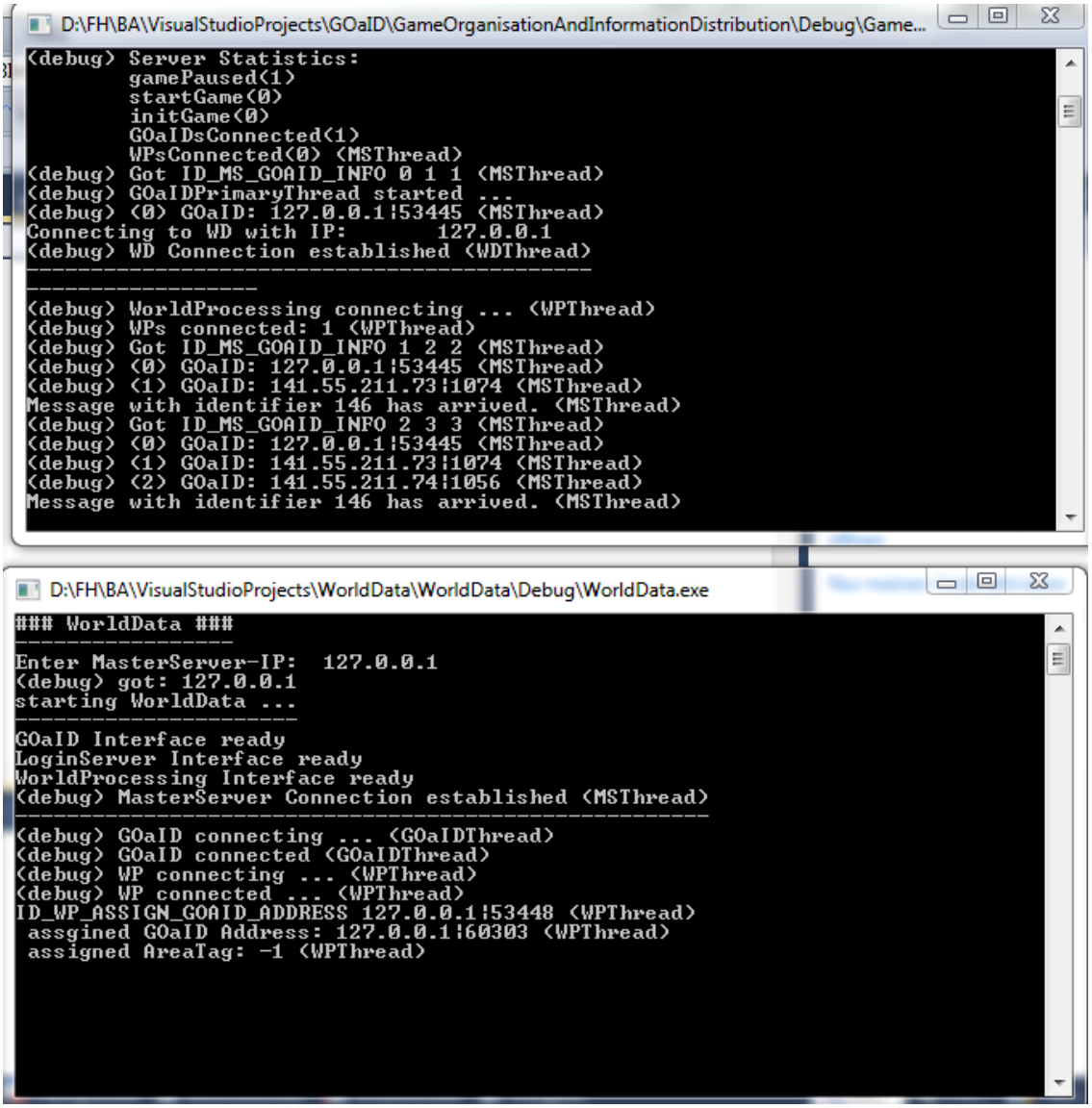

Abbildung B.2: Verbindungsaufbau der Komponenten GOaID und WorldWorldData

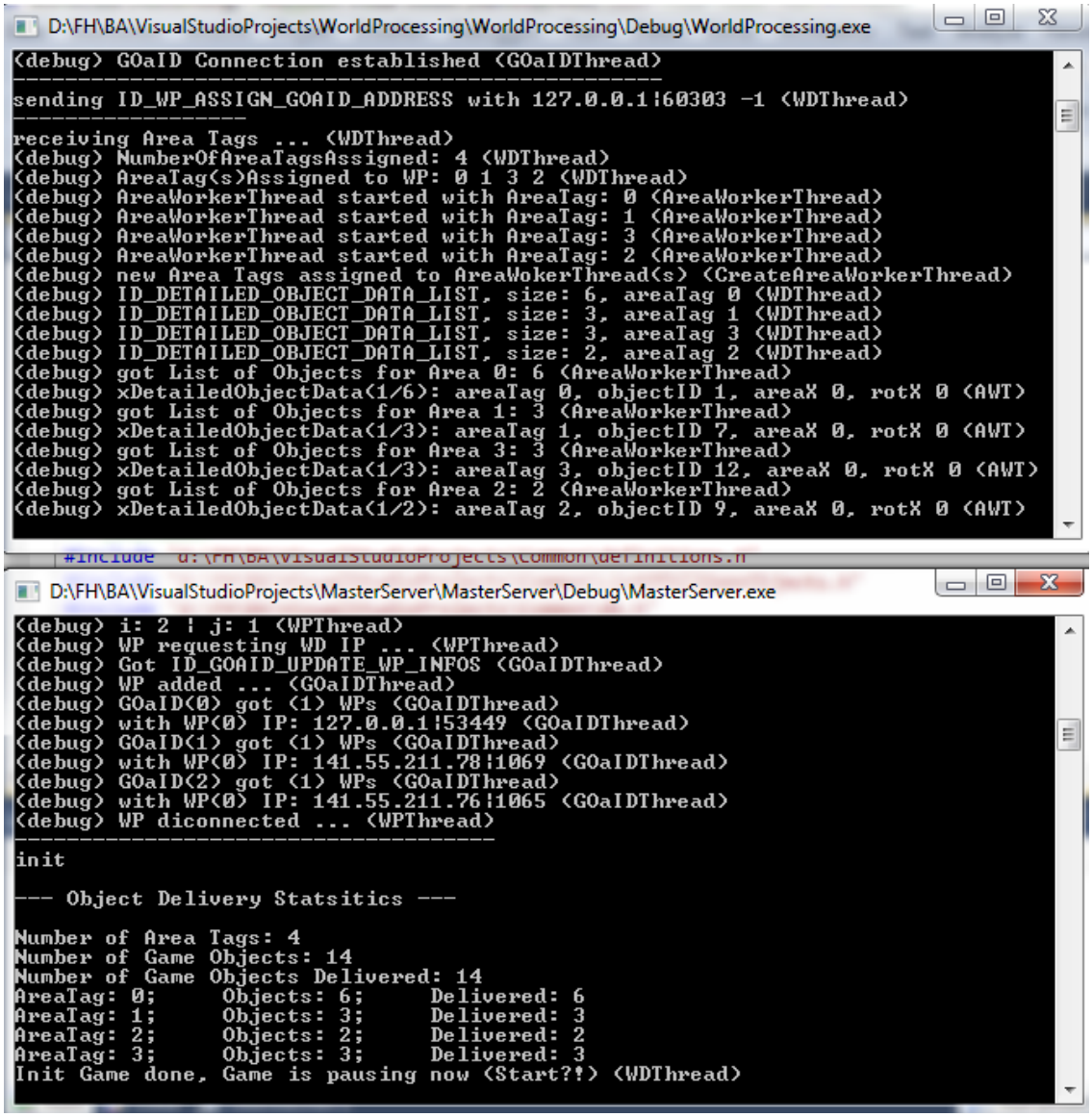

Abbildung B.3: Initialisierungsvorgang - Komponente MasterServer und WorldProcessing

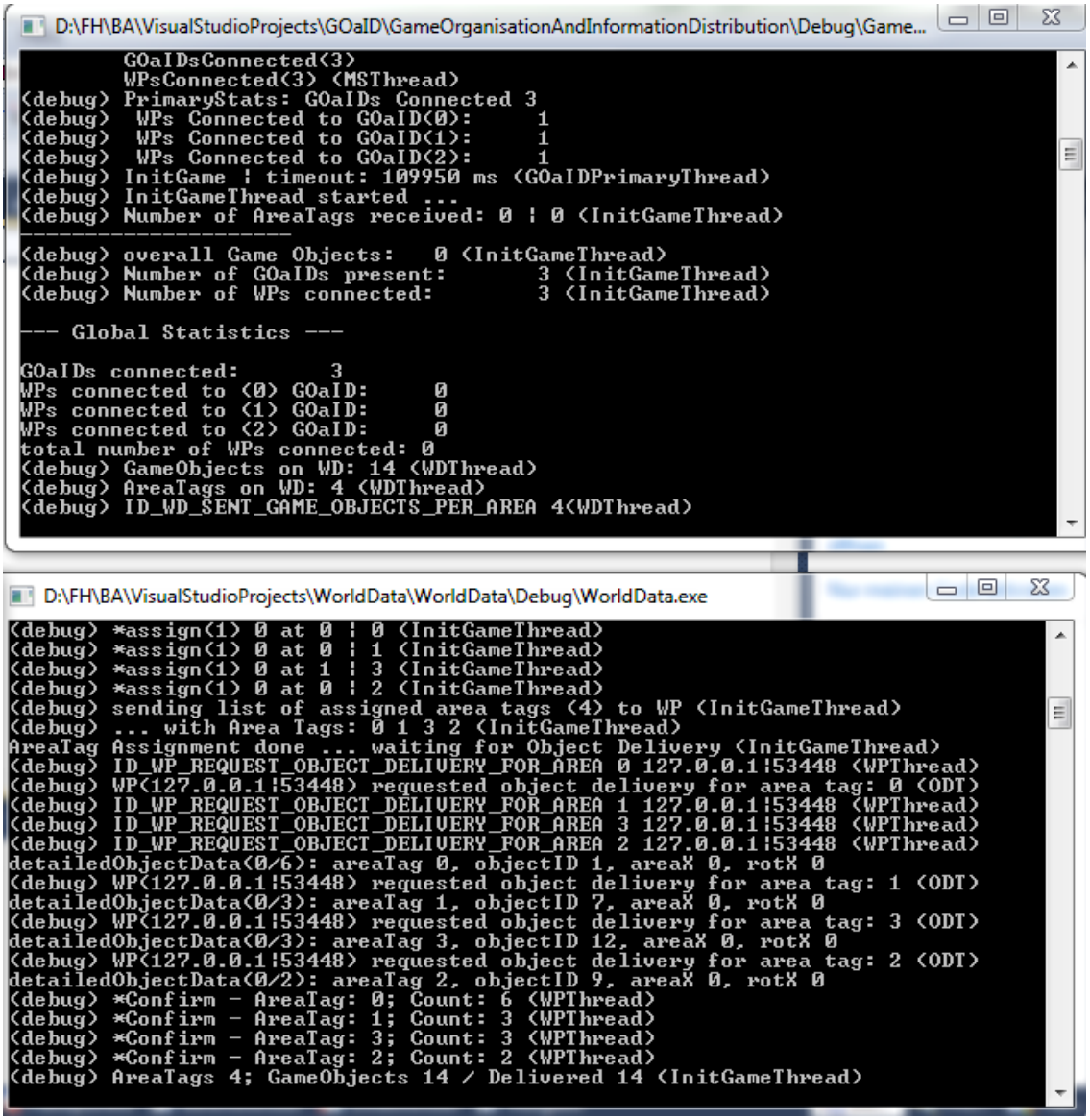

Abbildung B.4: Initialisierungsvorgang - Komponente GOaID und WorldData

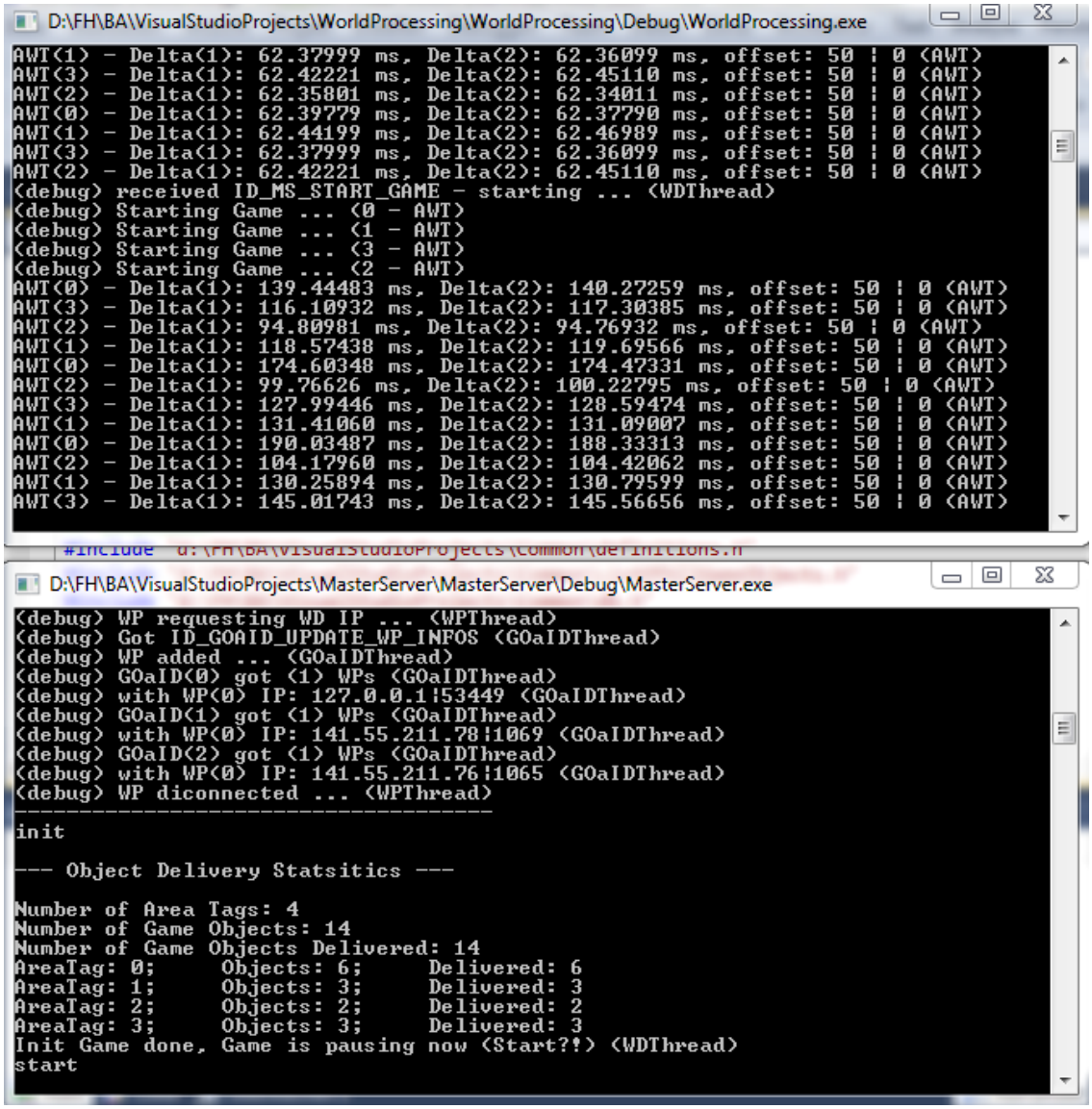

Abbildung B.5: Start des Systems - Komponente MasterServer und WorldProcessing

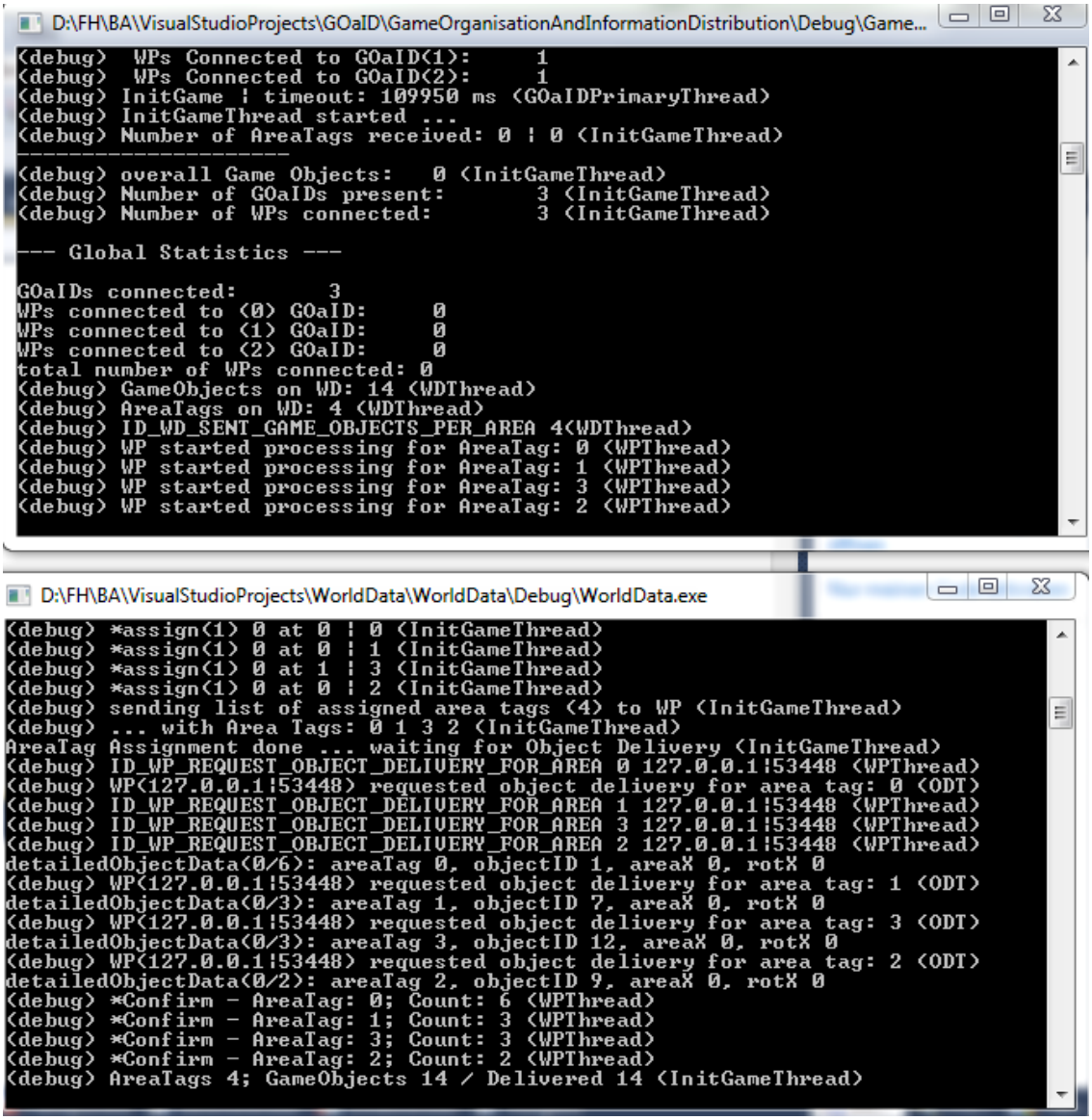

Abbildung B.6: Start des Systems - Komponente GOaID und WorldData

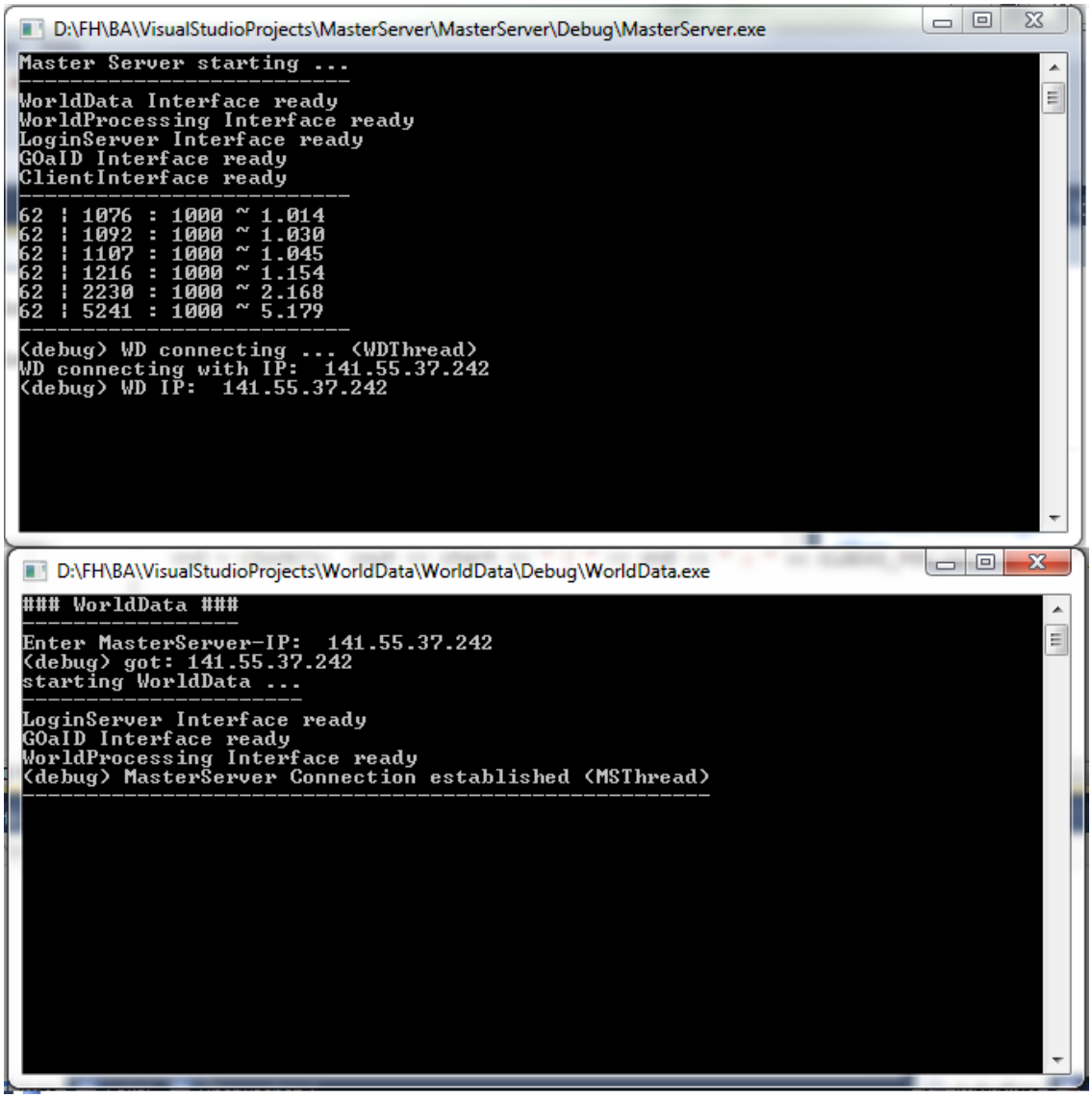

Abbildung B.7: Verbindungsaufbau von WorldData und MasterServer

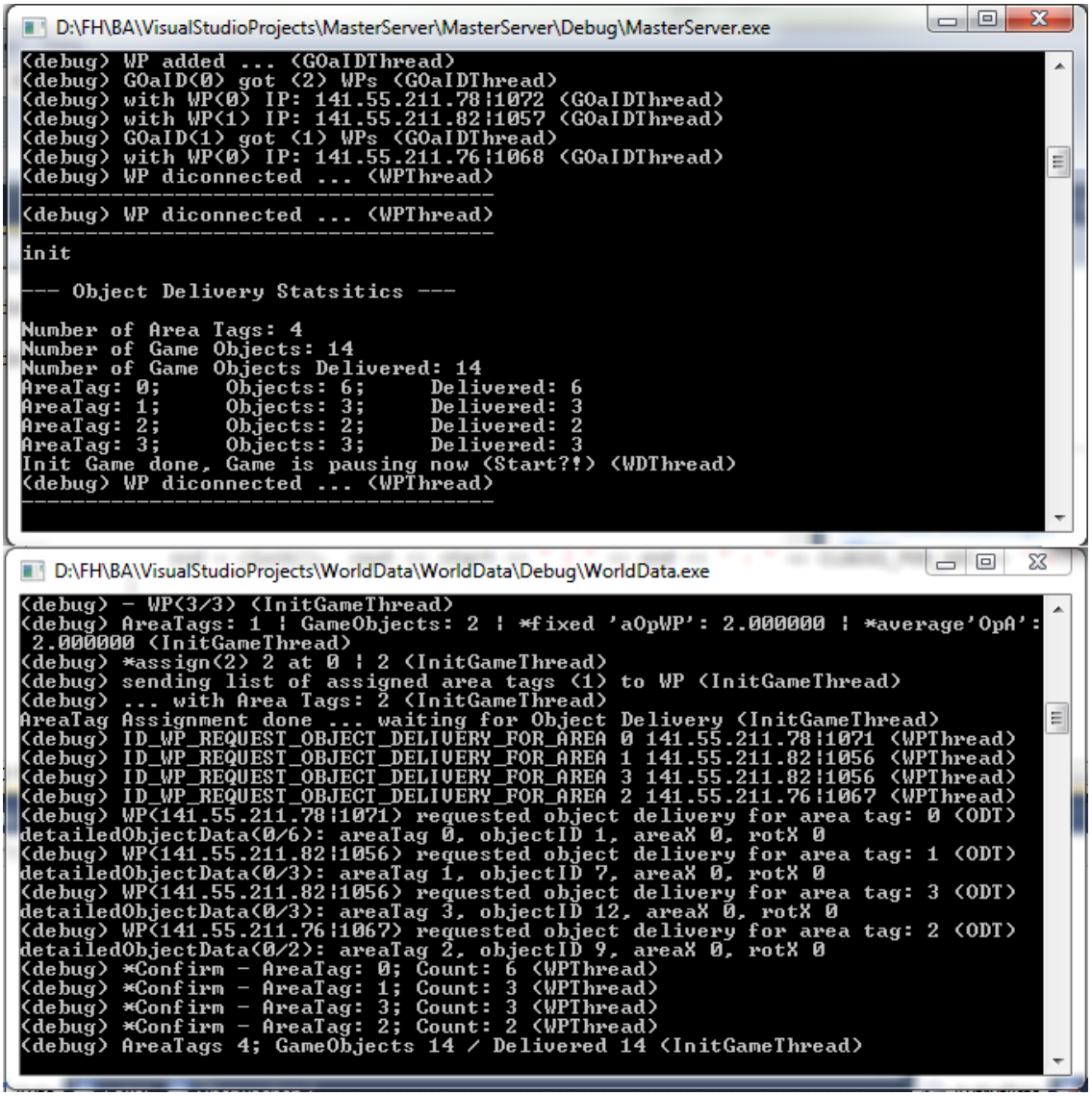

Abbildung B.8: Initialisierung verteilt über mehrere Rechner - MasterServer und WorldData Ansicht

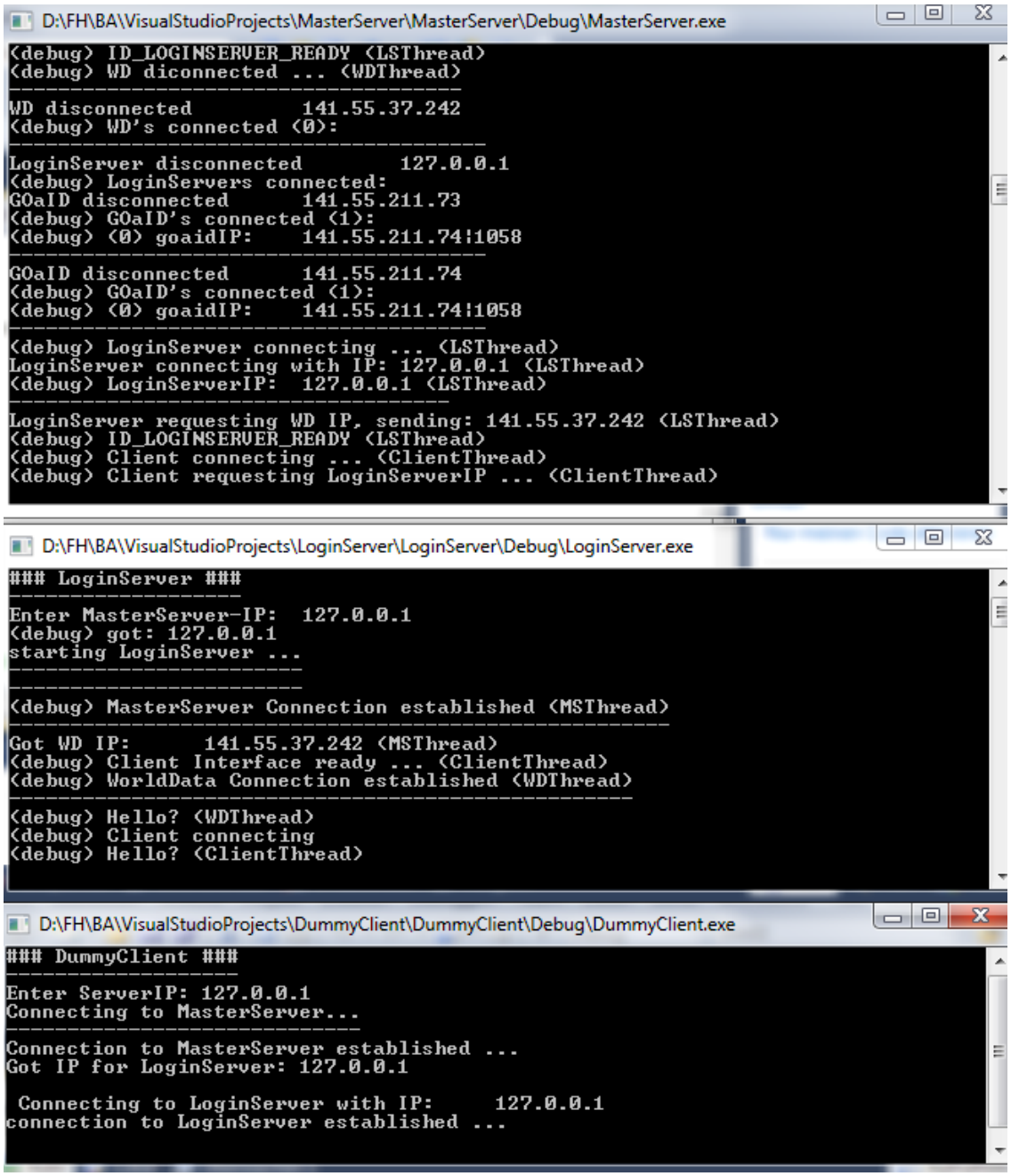

Abbildung B.9: Verbindungsweiterleitung des Clients zum LoginServer

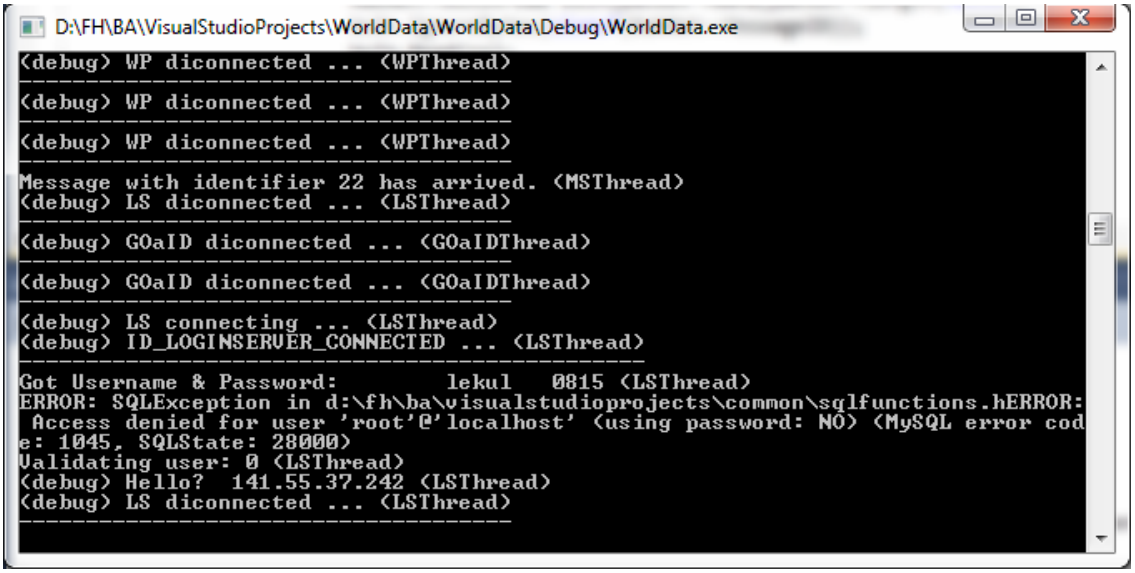

Abbildung B.10: Client Authentifizierung - WorldData Ansicht - Datenbank nicht erreichbar

Quellenverzeichnis

[1] Microsoft Developer Network: <http://msdn.microsoft.com/de-de/>, verfügbar am 11.08.2011

[2] RakNet Dokumentation: <http://www.jenkinssoftware.com/raknet/manual/index.html>, verfügbar am 11.08.2011

[3] MySQL Download-Seite: <http://www.mysql.de/downloads/>, verfügbar am 11.08.2011

[4] MySQL Connector Dokumentation: <http://dev.mysql.com/doc/refman/5.1/en/connector-cpp-apps-windows-visual-studio.html>, verfügbar am 11.08.2011

[5] MySQL Connector Codebeispiele: <http://dev.mysql.com/tech-resources/articles/mysql-connector-cpp.html>, verfügbar am 11.08.2011

[6] Stiegler, Andreas: MMO Server Structures: <http://www.hdm-stuttgart.de/~as147/mmo.pdf>, verfügbar am 11.08.2011

[7] Boost C++ Bibliotheken: <http://www.boost.org/>, verfügbar am 11.08.2011

[8] Egosoft Homepage, “X“ Spielreihe: <http://www.egosoft.com/>, verfügbar am 11.08.2011

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 15. August 2011