HTL – Perg

Höhere Abteilung für Informatik

DIPLOMARBEIT

Mine 4 Glory

Client – and serverbased game extensions

Projektteam: Alexander Ebner, Florian Fritzl

Projektbetreuer: Prof. Ing. Werner Schöller

Bearbeitungszeitraum: 01.10.2015 – 08.04.2016

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer, als der von uns angegebenen, Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg, _____ Unterschrift _____

Alexander Ebner

Perg, _____ Unterschrift _____

Florian Fritzl

Danksagung

Wir möchten uns bei allen Personen bedanken, welche uns während der Abwicklung dieser Diplomarbeit geholfen und unterstützt haben!

Besonderer Dank gilt unserem Betreuungslehrer Professor Ing. Werner Schöller, welcher uns vor allem bei der zeitlichen Planung und Umsetzung der Diplomarbeit geholfen hat, und uns jederzeit zur Besprechung des bisherigen Fortschritts unterstützend zur Seite gestanden ist.

Herzlichen Dank!

Zusammenfassung

Mine4Glory ist eine Diplomarbeit der Höheren Technischen Bundeslehranstalt für Informatik in Perg, die im Zuge der Matura im Jahr 2016 verfasst wurde.

Aufgabenstellung

Mine4Glory setzt sich aus 2 Teilbereichen zusammen, welche einerseits aus servererweiternden Plugins und andererseits aus clientbasierenden Modifikationen des Sandboxspiels Minecraft bestehen.

Realisierung

Am Client wird das Spiel um zahlreiche neue Kreaturen und Items erweitert, um dem einzelnen User mehr Spannung zu bieten.

Auf dem Server können mehrere Spieler, welche die gleiche Clientsoftware benutzen, in einem - auf dem Spielfilm „The Hunger Games“ basierenden - Spielmodus gegeneinander um ihr Überleben kämpfen.

Da die Diplomarbeit als Erweiterung für Minecraft gedacht ist, werden zum Benutzen der Erweiterungen lediglich das Originalspiel und eine Gratissoftware namens „Forge ModLoader“ zum Spielen benötigt.

Ergebnis

Das Ursprüngliche Spiel wurde clientseitig um vier Kreaturen, 21 Items und zwei Blöcke und serverseitig um ein Hub-, ein Gamelobby-, ein Extended Survival Games- und um ein Message - Plugin erweitert.

Abstract

Mine4Glory is a diploma thesis which was developed at the Higher Technical Institute for Informatics Perg in 2016.

Description

Mine4Glory is made up of two sub-parts, which are a server based extension called “plugin” and a client based modification. Both extensions are based on the sandbox game Minecraft.

Implementation

The client based modification adds new items, creatures and blocks to increase the excitement of the users.

On the server, several players which are using the same client software, can fight in a game mode based on the film “The Hunger Games” to be the last survivor.

Our diploma thesis is an extension for Minecraft, so to use our extension, the only things needed are the original game and a free to use software called „Forge Mod-Loader“.

Result

The basic game got extended on the client based-side by four creatures, 21 items and two blocks and on the server based side by a hub-, a game lobby-, an extended survival games- and a message – plugin.

Inhaltsverzeichnis

INHALTSVERZEICHNIS

1 Die Diplomanden

Alexander Ebner

Kontaktangaben

Steyrerstraße 60

4470 Enns

0660 586 82 47

alexanderebner97@gmail.com

Geburtsdatum: 06.12.1997

Ausbildung

2011 – 2016 HTL für Informatik Perg

2007 – 2011 HS Sankt Valentin

2003 – 2007 VS Sankt Valentin

Florian Fritzl

Kontaktangaben

Lanzenberg 72

4320 Perg

0681 203 085 32

florianfritzl@gmx.at

Geburtsdatum: 25.05.1997

Ausbildung

2011 – 2016 HTL für Informatik Perg

2007 – 2011 HS 1 Perg

2003 – 2007 VS Perg

2 Einleitung

Das Kapitel Einleitung beschäftigt sich mit der Entstehung, der Durchführung und den Zielen der Diplomarbeit. Nachfolgend werden diese Punkte detailliert beschrieben. Zu erwähnen ist, dass die gesamten Daten zu diesem Kapitel am Diplomarbeitsbeginn festgelegt wurden, während der Entwicklung berücksichtigt wurden und uns wesentlich dabei unterstützt haben, an unseren Zielen festzuhalten.

2.1 Entstehung und Ausgangssituation

Unsere Diplomarbeit basiert auf dem fertigen, jedoch vielseitig erweiterbaren, Sandbox - Game Minecraft.

“Minecraft is a game about breaking and placing blocks. At first, people built structures to protect against nocturnal monsters, but as the game grew players worked together to create wonderful, imaginative things.

It can also be about adventuring with friends or watching the sun rise over a blocky ocean. It’s pretty.”

[MOJANG SYNERGIES AB]

Minecraft genießt einen stetigen Zuwachs seiner Popularität, welcher auch im vierten Jahr nach der Erstveröffentlichung nicht abnimmt, eine gewaltig große Gemeinschaft von Spielern (von 2010 bis 2015 offiziell über 70 Millionen verkaufte Spiele für Windows, Linux, die Xbox Konsole und als Pocket Edition für das Mobiltelefon) und besitzt einerseits gute Dokumentation zum Entwickeln von Modifikationen und Plugins und andererseits auch eine große Community, welche sich mit Problemen in dieser Entwicklung auseinandersetzt.

Da sich die Diplomanden selbst mit dem Spiel intensiv beschäftigt und bereits Erfahrungen mit der Umgebung von Minecraft gemacht haben, scheint eine Diplomarbeit, welche an den Erfolg des Spiels anknüpft als hervorragend geeignet.

2014 wurde das Gründerunternehmen Mojang Synergies AB von Microsoft für 2,5 Milliarden US-Dollar gekauft. Dadurch änderte sich die gesetzliche Lage zur Entwicklung von Minecraft Erweiterungen. Auf die Legalität der Arbeiten im Zuge der Diplomschrift in Bezug auf Richtlinien von Microsoft wird im Kapitel 2.2 (Legalität) weiter eingegangen.

2.2 Legalität

2.2.1 Serverplugins

Grundsätzlich ist es nicht möglich, einen normalen Minecraft – Server, dessen Download auf der offiziellen Minecraft Website angeboten wird, durch Plugins zu erweitern. Jedoch gibt es verschiedene Anbieter (Bukkit, Spigot, Sponge, Cauldron), welche eine modifizierte Installationsdatei eines Minecraft – Servers anbieten. In unserem Fall wird die Software Spigot zur Installation des Servers und Bukkit zum Programmieren von den Plugins benutzt. Für Mojang war seit der ersten Veröffentlichung des Spiels eine schnelle Verbreitung am wichtigsten, und so wurden keine Einschränkungen in Bezug auf erweiterbare Server vorgenommen. Lediglich die Originalsoftware darf man nicht zu kommerziellen Zwecken nutzen, verbreiten oder anderen Personen unangemessener Weise Zugang verschaffen.

Abgesehen davon sind wir recht großzügig in Bezug auf Ihre Aktivitäten. Wir ermutigen Sie sogar dazu, spannende Dinge zu erstellen. Sie sollen nur das unterlassen, was wir Ihnen untersagen.

[MOJANG SYNERGIES AB]

Nach der Übernahme durch Microsoft wurden jedoch bereits kleine Einschränkungen für Serveranbieter vorgenommen. Nachdem Serveranbieter durch den Ingame-Verkauf von Rängen (Premium, VIP, …) deren Hardware und Softwarekomponenten finanzieren, müssen Spieler natürlich zu solchen Käufen ermutigt werden. Vor der Übernahme durch Microsoft konnten solchen ranghöheren Spielern zusätzliche Möglichkeiten, welche während dem Spielablauf eines Minispiels einen erheblichen Vorteil bringen, gegeben werden (Dieses Prinzip nennt man Pay-to-win). Nach dem Update des

Lizenzvertrages durch Microsoft wurde Pay-to-win verboten. Das bedeutet, dass Serveranbieter ranghöheren Spielern keinen Spielvorteil geben dürfen, nur weil diese für den Server bezahlen. Dadurch wurden die Möglichkeiten, einen Server zu finanzieren, erheblich eingeschränkt. Der Großteil der beliebten Servern bietet nun einzelne Minispiele nur noch für ranghöhere Spieler an (Pay-to-play) oder lässt ranghöhere Spieler rangniedere Spieler aus Runden werfen, wenn diese voll sind, damit ranghöhere Spieler auf Kosten der rangniederen jederzeit spielen können.

Wenn man sich also an diese Richtlinien hält, so ist das Anbieten eines durch Plugins erweiterten Servers durchaus erwünscht.

2.2.2 Clientmodifikation

Wenn das Originalspiel durch einen Kauf legal erworben wurde, darf es nach Belieben modifiziert werden. Die Weitergabe dieser modifizierten Version unterliegt aber bestimmten Richtlinien.

Der Begriff "Mod" bezeichnet ein Originalwerk von Ihnen oder jemand anderem, das keinen wesentlichen Teil unseres urheberrechtlich geschützten Codes oder Inhalts enthält. Wenn Sie Ihre Mod mit der Minecraft-Software kombinieren, nennen wir diese Kombination eine "modifizierte Version" des Spiels.

[MOJANG SYNERGIES AB]

Es ist nur die Weitergabe der Modifikation selbst gestattet, eine modifizierte Version von Minecraft weiterzugeben ist dagegen untersagt. Im Grunde ist Microsoft jedoch froh über die große Modding-Gemeinschaft, da die Firma dadurch einen großen Zuwachs interessierter Spieler bekommt.

2.3 Projektziel

Das Projektziel umfasst eine doppelseitige Erweiterung des Spiels Minecraft, bei welcher client- und serverseitige Features hinzugefügt werden. Im Detail sind diese am Client durch die Modifikation realisiert

- neue Waffen
- neue Rüstungen
- neue Kreaturen
- neue Blöcke
- neue Items.

Diese werden am Server durch Plugins an einem an der Filmtrilogie „Tribute von Panem - The Hunger Games“ angelehnten Spielmodus eingesetzt, wobei bis zu 24 Spieler gegeneinander ums Überleben kämpfen (Player vs. Player)

Folgende Grafik zeigt im blauen Bereich die von Microsoft vorgegebene Originalsoftware und im grünen Bereich die durch die Community gewünschten und in unserer Diplomarbeit erbrachten Erweiterungsmöglichkeiten

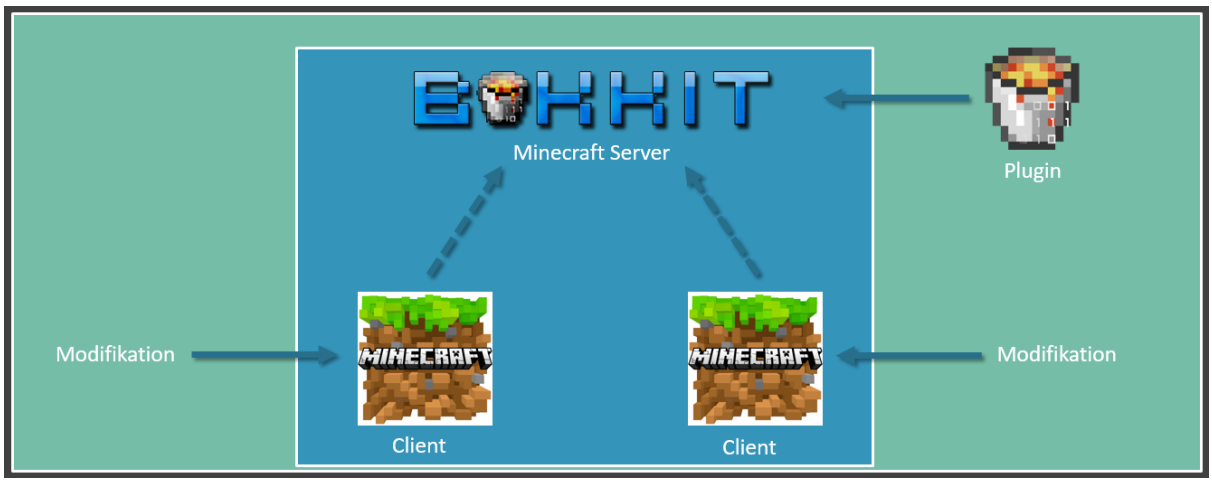


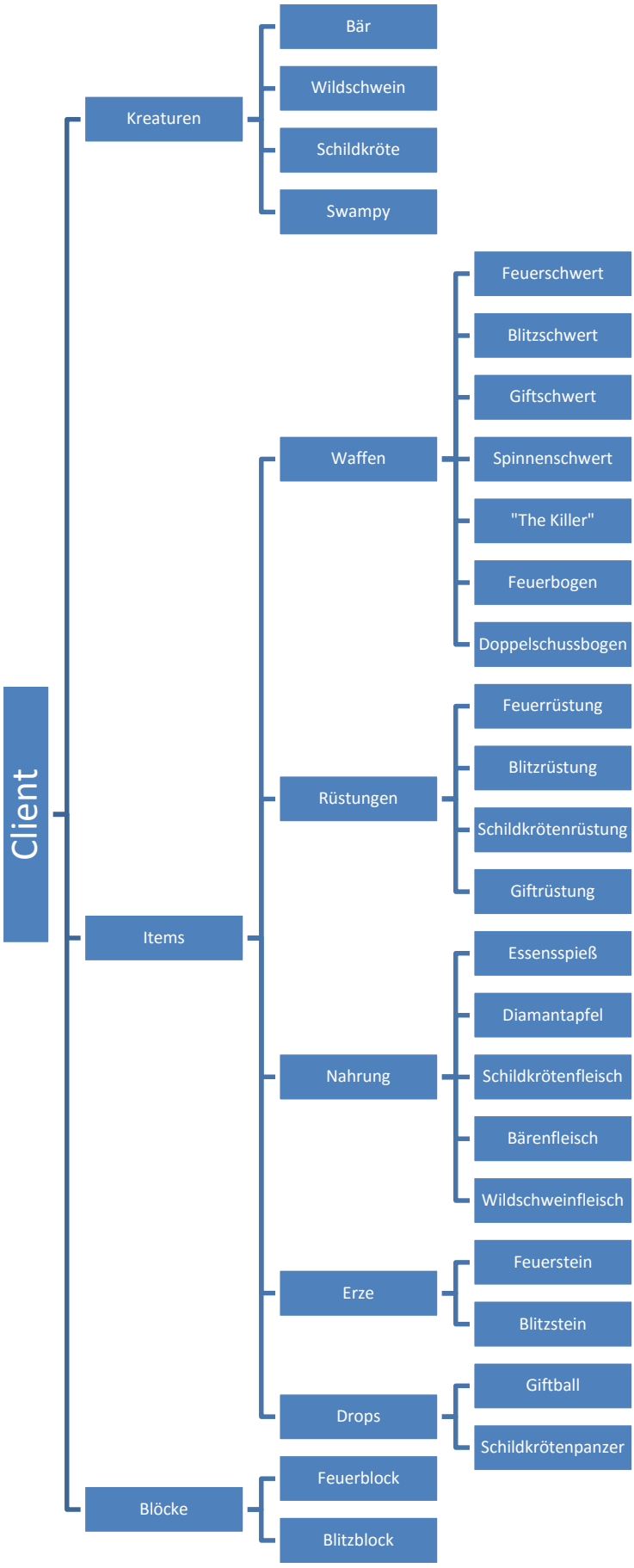
Abbildung 1: Übersicht Architektur

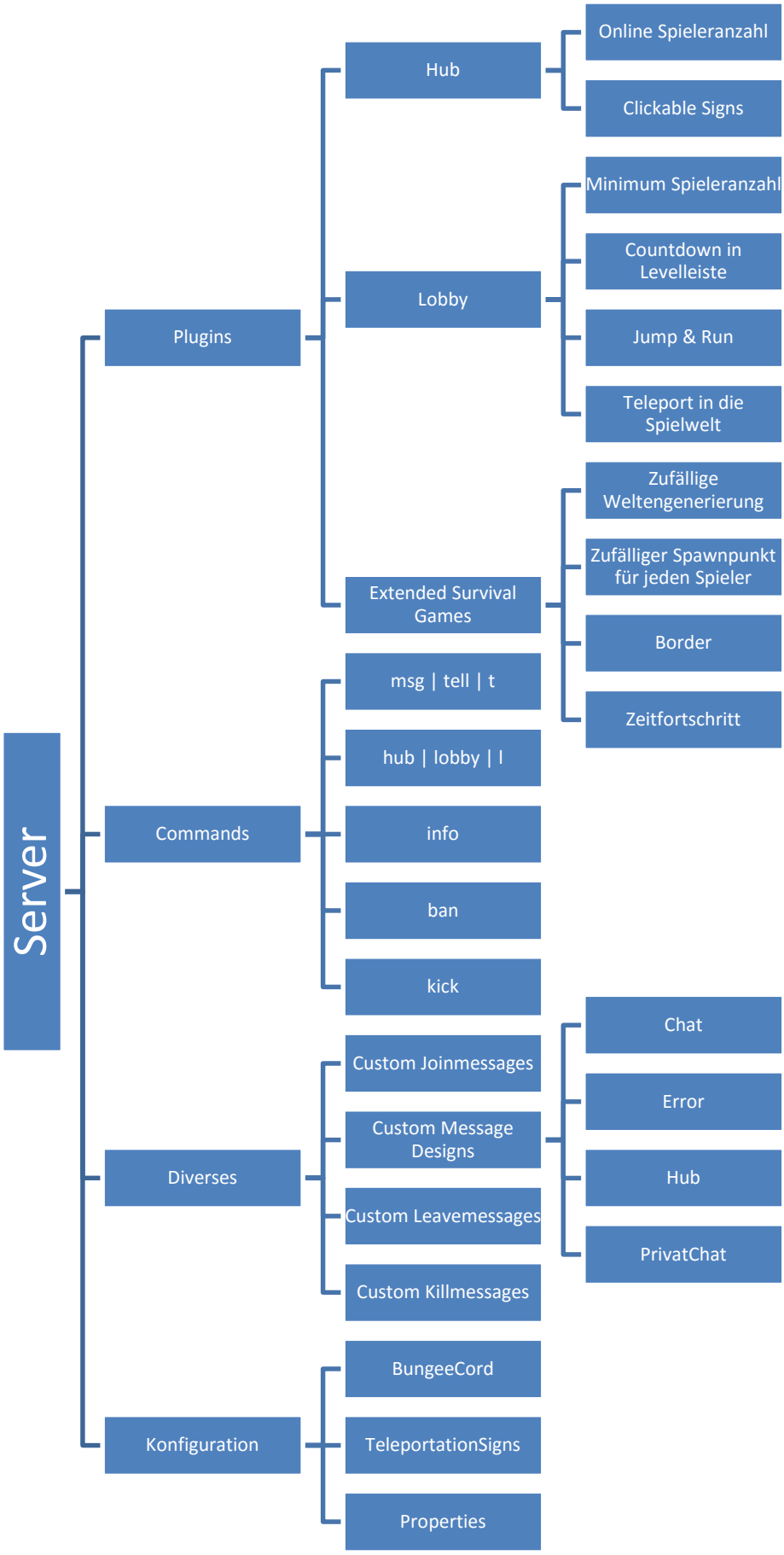
2.4 Geschäftsziel

Das Geschäftsziel der Diplomarbeit umfasst ein, durch die im Projektziel definierten Erweiterungen, problemloses Spielen auf dem Mine4Glory Minecraft Server, welcher durch das erweiterte Item- und Kreaturenangebot der Modifikation mehr Spaß und Abwechslung als herkömmliche Server bietet. Außerdem wird durch die Auseinandersetzung mit Entwicklungsumgebungen eine Erweiterung der Kenntnisse in der Programmierung mit Java und der 3D – Modellierung gewährleistet. Abhängig von der Popularität und Nutzung der Erweiterungen können am Server zukünftig auch Rangverkäufe getätigt werden, beziehungsweiße können durch die Downloadseite, auf der die Modifikation erhältlich ist, Werbungseinnahmen erzielt werden.

2.5 Diplomarbeitsstrukturpläne

Auf den nächsten 2 Seiten befinden sich die in die Clienterweiterung und Servererweiterung unterteilten Diplomarbeitsstrukturpläne.


Client
Kreaturen
Bär
Wildschwein
Schildkröte
Swampy
Items
Waffen
Feuerschwert
Blitzschwert
Giftschwert
Spinnenschwert
"The Killer"
Feuerbogen
Doppelschussbogen
Rüstungen
Feuerrüstung
Blitzrüstung
Schildkrötenrüstung
Giftrüstung
Nahrung
Essensspieß
Diamantapfel
Schildkrötenfleisch
Bärenfleisch
Wildschweinfleisch
Erze
Feuerstein
Blitzstein
Drops
Giftball
Schildkrötenpanzer
Blöcke
Feuerblock
Blitzblock


Server
Plugins
Hub
Online Spieleranzahl
Clickable Signs
Lobby
Minimum Spieleranzahl
Countdown in Levelleiste
Jump & Run
Teleport in die Spielwelt
Extended Survival Games
Zufällige Weltengenerierung
Zufälliger Spawnpunkt für jeden Spieler
Border
Zeitfortschritt
Commands
msg | tell | t
hub | lobby | l
info
ban
kick
Diverses
Custom Joinmessages
Custom Message Designs
Chat
Error
Hub
PrivatChat
Custom Leavemessages
Custom Killmessages
Konfiguration
BungeeCord
TeleportationSigns
Properties

2.6 Grenzkriterien

Nachfolgend finden sich die Wunsch- und Abgrenzungskriterien. Die Wunschkriterien sind essenziell für die Erreichung des Ziels der Diplomarbeit. Die Abgrenzungskriterien sind Ziele, welche zwar zu erreichen wären, jedoch einen zu hohen Aufwand benötigen würden um sie im Zuge der Diplomarbeit zu verwirklichen.

2.6.1 Wunschkriterien

Das wichtigste Kriterium der Diplomarbeit ist es, die doppelseitige Erweiterung fehlerfrei zu implementieren. Außerdem soll die Modifikation am Server installiert sein und Spieler sollen diese gemeinsam benutzen können, um im Spielmodus mit den durch die Modifikation hinzugefügten Features gegeneinander zu kämpfen.

2.6.2 Abgrenzungskriterien

Die folgenden Abgrenzungskriterien schießen über das Ziel hinaus und werden daher im Projekt nicht berücksichtigt. Warum manche dieser Kriterien, welche am Projektbeginn ein Teil des Projektziels waren während des Entwicklungsprozesses zu Abgrenzungskriterien geändert wurden, wird im Kapitel 7 (Probleme und Zukunftsaussichten) genauer erklärt.

- Kommandozeilenbefehle für das Serverplugin
  - /friend
  - /party
  - /premium
  - /start
  - /chatlog
  - /reply
- Datenbankanbindung für das Serverplugin
- Premiumverkauf für das Serverplugin
- Kein Support für Minecraft Versionen, welche nicht 1.8 kompatibel sind
- Keine Veränderung der Weltengeneration
- Kein Einfügen von Kreaturen, welche ihre Gestalt verändern können.

2.7 Zielgruppen

Minecraft spricht größtenteils männliche Spieler unter 18 Jahren an. Auf eben diese Zielgruppe zielt unsere Diplomarbeit ab, jedoch spricht im Gegensatz zur Modifikation, welche in deutscher und englischer Sprache angeboten wird, der Server nur den deutschsprachigen Raum an. Dies ist die Folge eines Problems, welches später noch in Kapitel 7 (Probleme und Zukunftsaussichten) erläutert wird.

2.8 Projektablauf

Meilenstein	Soll-Datum	Ist-Datum
Konfiguration des Bukkit-Servers abgeschlossen	19.10.2015	10.10.2015
Pflichtenheft fertiggestellt	26.10.2015	26.10.2015
Entwicklungsumgebungen eingerichtet	14.09.2015	30.11.2015
Modimplementierung fertiggestellt	11.01.2016	22.02.2016
Pluginimplementierung fertiggestellt	11.01.2016	26.02.2016
Kommunikation zwischen Mod und Plugin hergestellt	08.02.2016	05.02.2016
Diplomschrift fertiggestellt und gedruckt abgegeben	08.04.2016	08.04.2016

3 Ressourcenplanung

Im Kapitel Ressourcenplanung werden alle Ressourcen dargestellt, welche zur Erstellung der Diplomarbeit von Nöten waren. Dazu gehört neben Hard- und Software auch das Projektteam. Den Schluss dieses Kapitels bildet eine Aufwandsschätzung, bei der versucht wurde, unsere Zeit grob in Aufgaben zu unterteilen.

3.1 Hardware

Dieses Kapitel befasst sich mit jeglicher Hardware, die während der Diplomarbeit eingesetzt wurde. Dies ist einerseits die Entwicklungshardware der Schüler und andererseits der Schulserver. Nachfolgend findet sich eine detaillierte Darstellung der Baukomponenten unserer Hardware.

3.1.1 Hardware des Schulservers

- Betriebssystem: Microsoft Windows Server 2012
- 64-Bit Betriebssystem
- Host: hyperion.main.htl-perg.at
- CPU: 4vCPU
- 6 GB Arbeitsspeicher

3.1.2 Standrechner Fritzl

- Betriebssystem Windows 10
- CPU: Intel Core i5 – 4440 4x 3,3 GHz
- 8 GB Arbeitsspeicher
- 64-Bit Betriebssystem
- 1 TB Festplatte + 250 GB SSD

3.1.3 Laptop Ebner

- Betriebssystem Windows 10
- CPU: Intel Core i5 – 5200U 2x 2,2 GHz
- 8 GB Arbeitsspeicher
- 64 Bit Betriebssystem
- 250 GB SSD

3.2 Software

Die Software als Ressource zur Entwicklung der Diplomarbeit wird in Kapitel 4 (Entwicklungssysteme) erläutert.

3.3 Projektteam

Die Diplomarbeit wird in der Regel in Teamarbeit durchgeführt (Richtwert für die Größe des Projektteams: 2 bis 5 Personen) und ist eine in sich geschlossene Arbeit.

[PERG, HTL, 2015]

Nach diesen offiziellen Angaben, der HTL Perg Website entnommen, formte sich das Projektteam aus Alexander Ebner und Florian Fritzl, welche sich mit der Diplomarbeit nun 150-180 Stunden parallel laufend zum täglichen Schulrhythmus beschäftigten. Es wird also eine Gesamtstundenanzahl von etwa 300-360 vorgegeben.

3.4 Aufwandsschätzung

Folgend findet sich die am Beginn der Diplomarbeit angefertigte Aufwandsschätzung. Dabei wurde versucht, die vorgegebene Stundenanzahl in kleinere Pakete einzuteilen, um während der Entwicklung zu bemerken, wie effizient wir unsere Stunden verwendeten, ob wir in der Zeit lagen und ob die Ziele, die wir uns setzten zu erreichen waren.

Tätigkeit	Anzahl Stunden (340)
Projektplanung	**60**
Clientmod	**90**
Planung	10
Implementation	70
Testen	10
Serverplugin	**90**
Planung	15
Implementation	60
Testen	15
Diplomschrift	**100**

4 Entwicklungssysteme

Das Kapitel Entwicklungssysteme befasst sich mit der Software als Ressource zur Erstellung der Diplomarbeit. Nachfolgend sind lediglich jene Programme beschrieben, welche nicht nur zeitlich am meisten benutzt wurden, sondern auch den größten Faktor zur reibungslosen Arbeit an der Diplomarbeit stellten und somit unersetzbar waren. Auch wurde alltägliche Bürosoftware nicht beschrieben.

4.1 Windows Server 2012 R2

Die Software Windows Server 2012 R2 wurde von Microsoft im Jahr 2012 veröffentlicht. Nachdem im Schulunterricht bereits mit dieser Software gearbeitet wurde, schien sie hervorragend dazu geeignet zu sein, diese zum Hosten des Minecraft Servers zu verwenden.

Der Windows Server 2012 R2 dient der Diplomarbeit, indem er Services wie den BungeeCord Bukkit Minecraft Server hostet. Der Zugriff auf den Server erfolgt einerseits schulintern und andererseits auch außerhalb des Schulnetzes via VPN. Extern wurde der Port 25565 (Minecraft-Standardport) freigeschalten, auf welchem die Minecraft Clients über das Web (ohne VPN Verbindung zur Schule) den Server erreichen können.

Sobald der Client den Server am Port 25565 erreicht, kann dieser via BungeeCord den Client an andere Minecraft Server, welche auf diversen anderen Ports laufen weiterleiten. Dazu aber mehr in Kapitel 4.2 (BungeeCord Bukkit Minecraft Server).

4.2 BungeeCord Bukkit Minecraft Server

Der BungeeCord Bukkit Minecraft Server besteht aus mehreren einzelnen Bukkit Minecraft Servern. Jeder einzelne dieser Bukkit Minecraft Server ist ein Minecraft Server, welcher Plugins unterstützt. Nachdem der Standardport für Minecraft (25565) bereits durch BungeeCord belegt wird, um alle Logins von Clients zu kontrollieren und um dann den Spieler auf die Main-Lobby weiterzuleiten, müssen die einzelnen Server jeweils diverse andere Ports verwenden. Obwohl vom Windows Server 2012 R2 nur der eine Port nach außen hin freigegeben ist, kann BungeeCord die Spieler auf die spezifischen Ports der von BungeeCord verwalteten Server weiterleiten, und damit die Spieler zu anderen Servern senden. Die Software BungeeCord vereint dabei Server, indem sie alle Ports der einzelnen Server kennt.

Ein positiver Aspekt dabei ist, dass der Endbenutzer selbst keinerlei Veränderungen an seinem Client bemerkt, da die Überbrückung auf einen anderen Server einem normalen Teleport gleicht.

Im Zuge der Diplomarbeit wurde die Software dazu genutzt, drei einzelne Minecraft Server, welche aus der Main-Lobby, der Spielelobby und dem Spielserver selbst bestehen, zu verbinden

Die Konfiguration vom BungeeCord Server wird in Kapitel 5.2 (BungeeCord Bukkit Minecraft Server) genauer dargestellt.

4.3 Eclipse

Die integrierte Entwicklungsumgebung (IDE) Eclipse wurde im Zuge der Diplomarbeit zur Entwicklung jeglicher Software, welche später im Serverbereich als Plugin und im Clientbereich als Modifikation verwendet wurde, herangezogen

Zur Verwendung der IDE mussten zu Beginn der Diplomarbeit zahlreiche Konfigurationen vorgenommen werden, um diese überhaupt nutzen zu können, beziehungsweise um dann auch effizient und benutzerfreundlich damit arbeiten zu können.

Da Eclipse nicht die einzige Entwicklungsumgebung ist welche die Programmierung in Java anbietet, musste eine Entscheidung gefällt werden, warum genau Eclipse verwendet werden soll.

4.3.1 Eclipse für die Serverprogrammierung

Bei Eclipse für die Serverprogrammierung gab es folgende Kriterien, welche die IDE von den Konkurrenten abhob:

- Vollständige Installationsanleitung der Konfiguration von Eclipse für Bukkit Developer
- M2Eclipse – Erweiterung und Maven in Eclipse integriert zu haben
- Dokumentation und Tutorials fast ausschließlich in Eclipse
- Die IDE wurde bereits im Schulunterricht herangezogen, um Java Projekte zu programmieren, somit kannte man sich mit dem User Interface schon aus.

4.3.2 Eclipse für die Clientprogrammierung

Bei Eclipse als IDE für die Clientprogrammierung gab es folgende Kriterien, welche die IDE von den Konkurrenten abhob:

- Eclipse ist die Haupt-Entwicklungsumgebung der Forge Entwickler, deshalb gibt es für Eclipse den besten Support.
- Dokumentation und Tutorials fast ausschließlich in Eclipse
- Installation von Forge ist auf Eclipse abgestimmt
- Die IDE wurde bereits im Schulunterricht herangezogen, um Java Projekte zu programmieren, somit kannte man sich mit dem User Interface schon aus.

4.4 Dropbox

Dropbox ist ein Online Cloud-Speicher. Daher ermöglicht die Software, Dateien online zu speichern, zu ändern und zu löschen. Im Zuge der Diplomarbeit wurde ein Dropbox Ordner erstellt, auf den lediglich die zwei Diplomanden zugreifen können.

Die Software wurde benutzt, um Dokumente, die während der Arbeit anfielen, zu teilen und um sofort beiden Diplomanten Zugriff darauf zu geben.

Beispiele hierfür sind:

- Zeitplanungen
- Projektpräsentationen
- Projektziele, Geschäftsziele
- Die Diplomschrift
- Stundentafeln
- Code

Doch nicht nur das Teilen von Dokumenten zwischen den Diplomanden war ein großer Vorteil, welchen uns Dropbox gewährleistet, es war auch die Kommunikation zum Server. Denn jedes Mal, wenn man ein Plugin um Code veränderte, musste man dieses am Server neu einspielen. Und dies funktionierte durch Dropbox. Durch einfaches Speichern der Plugin-Datei synchronisierte Dropbox automatisch die Datei auf den Server, und so war lediglich ein Neustart des Servers erforderlich, um alle Änderungen sofort wirksam zu machen.

4.5 Techne

Techne ist ein Tool, welches erlaubt, komplizierte Modelle von Kreaturen für Minecraft zu konstruieren. Mithilfe dieses Tools wird das modellieren von Kreaturen enorm erleichtert. Auch das Einbinden in den Code wird verbessert, da dieses Programm eine externe Datei mit allen Koordinaten und Rotationen, welche zur Darstellung einer Animation von Nöten sind, zur Verfügung stellt. Im Zuge der Diplomarbeit wurde es verwendet, um 3D Modelle der Kreaturen zu erstellen.

4.5.1 Funktionen

Dimensions

Erweitert oder verringert den ausgewählten Quader an der X-, Y- oder Z-Achse.

Position

Verschiebt den ausgewählten Quader an der X-, Y- oder Z-Achse.

Offset

Verschiebt den Quader relativ zum Rotationspunkt, der Rotationspunkt bleibt dabei auf derselben Position.

Texture Offset

Verschiebt die Felder für die Texturen, damit wird die Überlappung einzelner Felder verhindert.

Rotation

Damit kann der Quader um die X-, Y- oder Z-Achse rotiert werden.

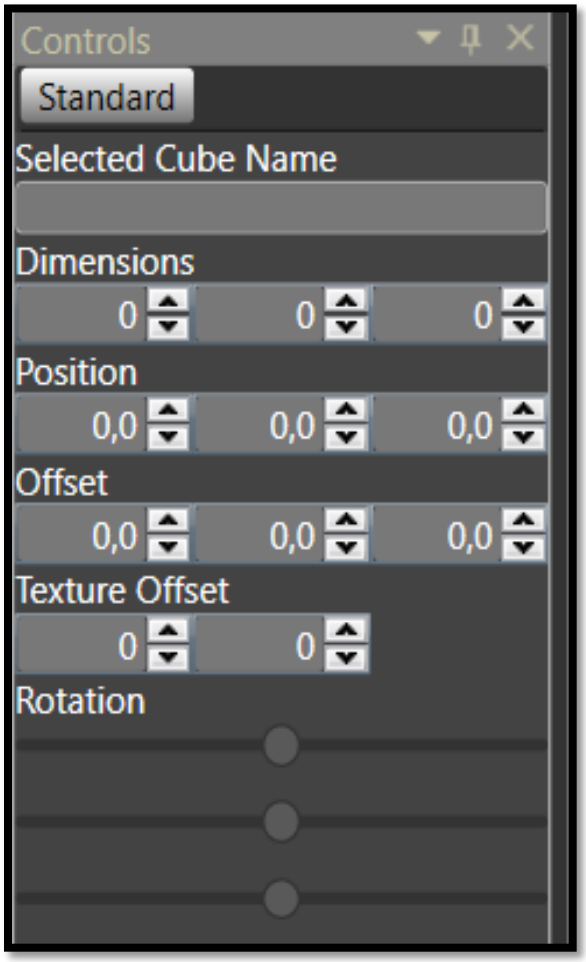


Abbildung 2: Techne Menü

Exportieren als Texturemap

Damit wird das Modell als Texturemap exportiert (PNG-File) und kann mit einem beliebigen Grafiktool designet werden.

4.5.2 Kreaturen

Vorbereitung des Modells

Der erste Punkt des Modellierens ist das Erstellen eines Modells, auf dem die zu erstellende Kreatur aufbaut.

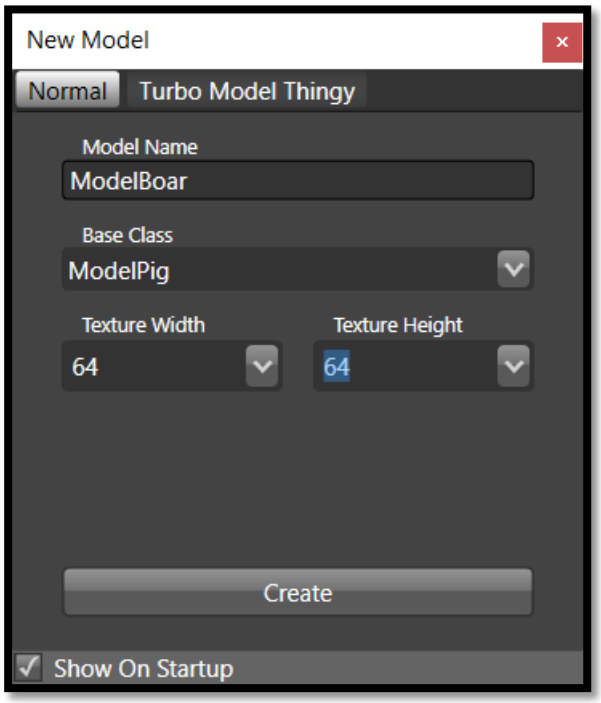


Abbildung 4: Neues Modell erstellen

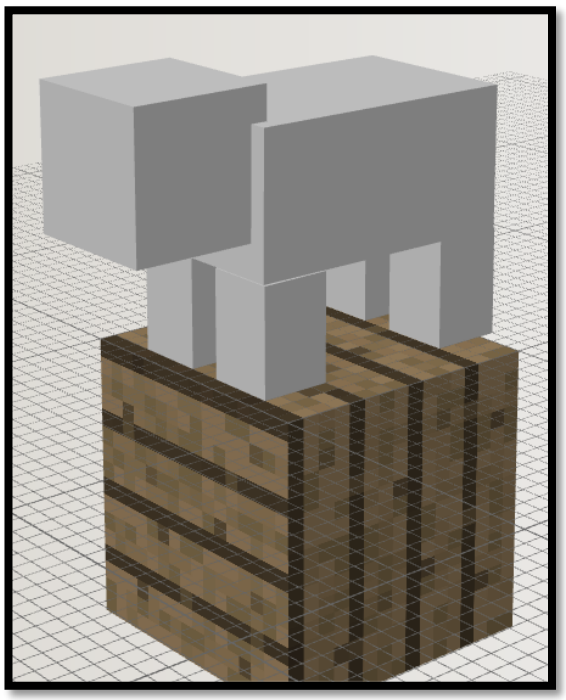


Abbildung 3: Basismodell des Schweins

Wie man an diesem Beispiel sieht, wird ein Modell eines Schweins als Vorlage benutzt. Auf diesem Modell baut das modifizierte Modell eines Wildschweins auf.

Verändern des Modells

Es werden verschiedene Bereiche des Modells verkleinert, vergrößert oder rotiert. Es können aber auch komplett neue Teile hinzugefügt werden.

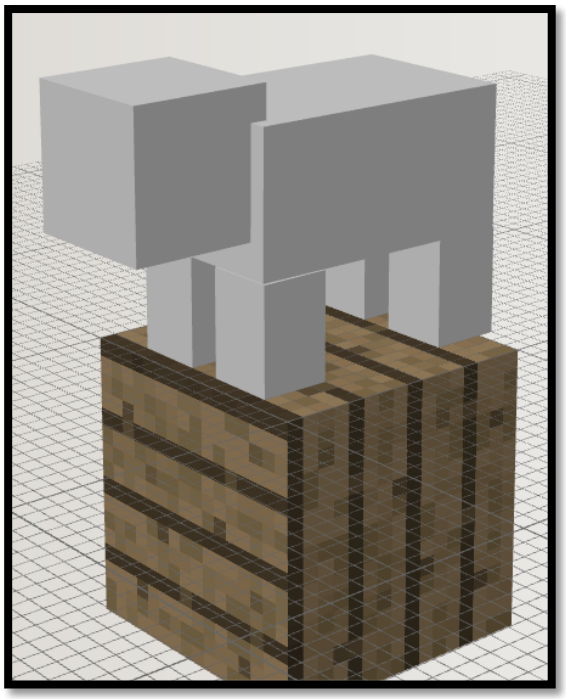


Abbildung 6: Wildschweinmodell vorher

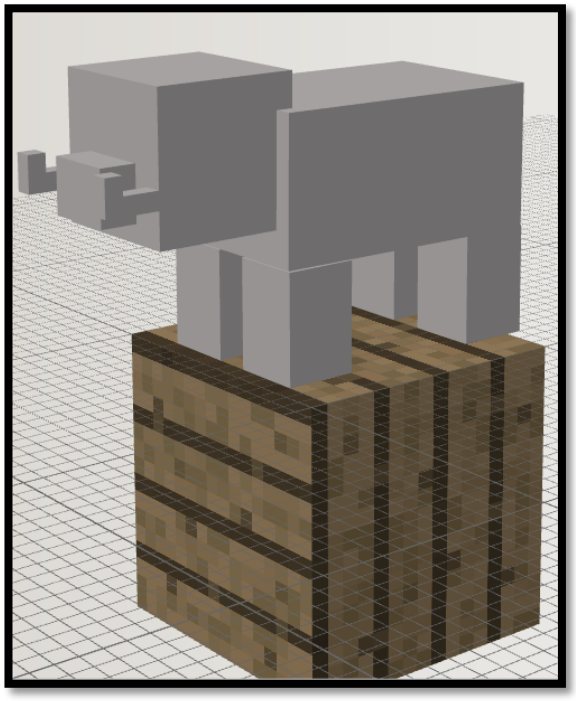


Abbildung 5: Wildschweinmodell nachher

In diesem Bild erkennt man die Stoßzähne und die Schnauze, die dem Modell des Schweines hinzugefügt wurden. Außerdem wurde das gesamte Modell um den Faktor 1.1 vergrößert.

Exportieren als Texturemap

Um das nun erstellte Modell der Kreatur zu designen, muss zuerst eine Texturemap exportiert werden.

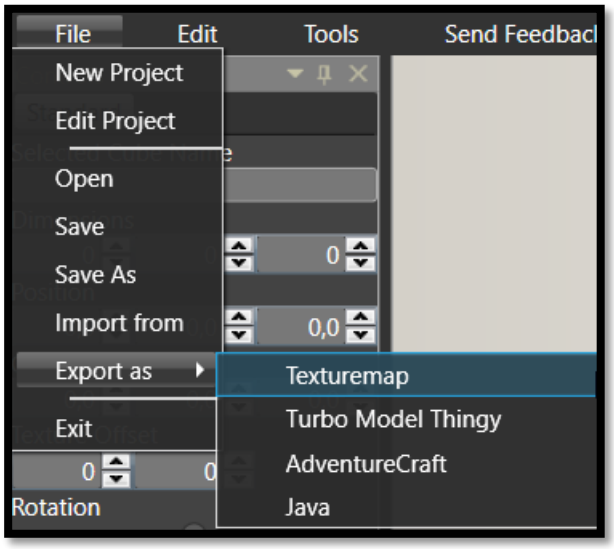


Abbildung 7: Exportieren als Texturemap

Verändern der Texturemap - Designen der Kreatur

Die vorher exportiere Texturemap wird nun mit dem Bildbearbeitungstool GIMP geändert. Die Funktion von GIMP findet sich detailliert in Kapitel 4.6 (GIMP)

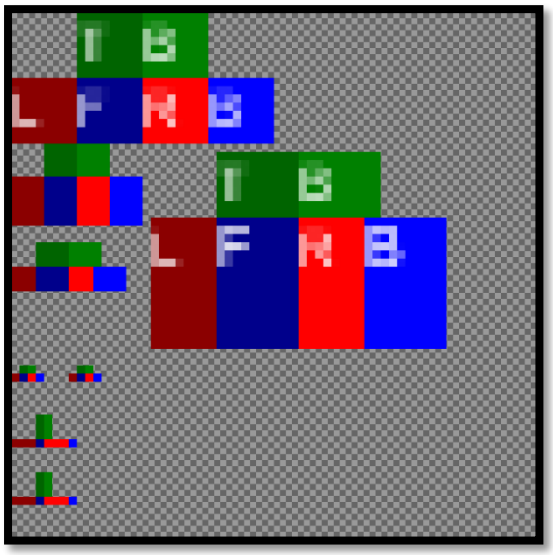


Abbildung 9: Vorlage Texturemap Wildschwein

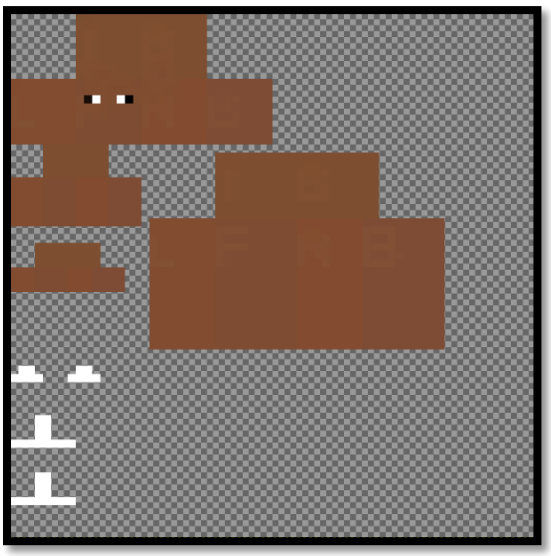


Abbildung 8: Eingefärbte Vorlage Texturemap Wildschwein

Einfügen der Texturemap in Techne

Die fertig eingefärbten Texturen können nun als PNG - Dateien abgespeichert werden und können danach wieder von Techne importiert werden.

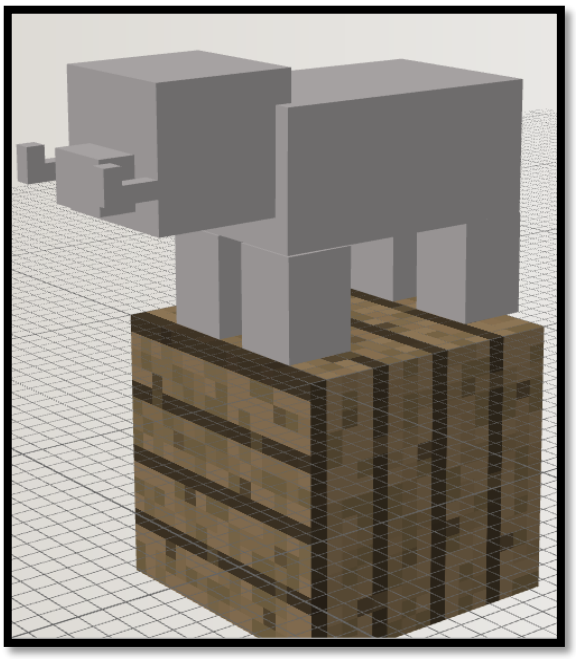


Abbildung 11: Fertiges Modell Techne

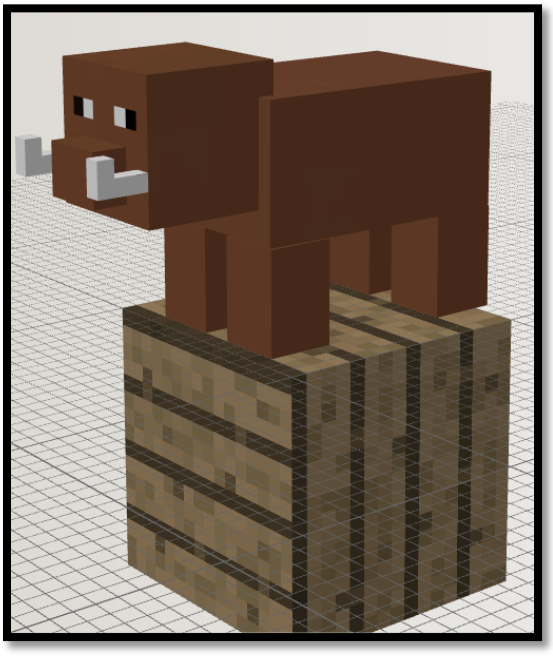


Abbildung 10: Fertiges Modell mit importierter Texturemap

Exportieren als Java-Code

Um das kreierte Modell im Spiel selbst anzeigen zu können, werden die Koordinaten, die Größe und die Positionen der einzelnen „Boxen“, die im Endeffekt das Modell ergeben, benötigt. Diese kann man aus Techne exportieren und später in das Programm implementieren.

```
head = new ModelRenderer(this, 0, 0);
head.addBox(-4F, -4F, -8F, 8, 8, 8);
head.setRotationPoint(0F, 12F, -6F);
head.setTextureSize(64, 64);
head.mirror = true;
setRotation(head, 0F, 0F, 0F);
```

Abbildung 12: Codebeispiel Techneexport

Diese Attribute werden für alle „Boxen“ erstellt, können aber nicht auf genau diese Art in den bestehenden Code eingefügt werden.

Wie man im nachfolgenden Beispiel sieht, wurden dem „head“ Attribut drei Boxen hinzugefügt, diese stehen für die beiden Hörner und die Schnauze.

```
this.head = new ModelRenderer(this, 0, 0);
this.head.addBox(-4F, -4F, -6F, 8, 8, 6);
this.head.setRotationPoint(0F, 4F, -8F);
this.head.setTextureOffset(22, 0).addBox(-5F, -6F, -4F, 2, 3, 1, 0.0F);
this.head.setTextureOffset(22, 0).addBox(3F, -6F, -4F, 2, 3, 1, 0.0F);
this.head.setTextureOffset(22, 0).addBox(-1F, 0F, -8F, 2, 2, 2, 0.0F);
```

Abbildung 13: Codebeispiel Implementation aus Techne

4.6 GIMP

GIMP ist ein Programm zur Bearbeitung von Bildern. Es bietet eine Vielzahl an Funktionen und ist als beliebtestes Gratis-Bildbearbeitungsprogramm ein sehr guter Ersatz für das professionelle Programm Photoshop. Im Zuge der Diplomarbeit wurde es zum Designen der 3D Kreaturen verwendet.

Die in diesem Programm erschaffenen Items wurden in GIMP erstellt und designet. Dafür wurde meistens ein bereits bestehendes Item, welches dem zu erstellendem Item ähnelt, als Basis verwendet. Dieses wurde dann in seiner Form und Farbe so verändert, dass es den jeweiligen Anforderungen an das Item entspricht. Dafür wurden hauptsächlich die Funktionen „Nach Farbe auswählen“, „Einfärben“ und „Pinsel“ verwendet.

Abbildung 14: Gimp GUI

5 Konfiguration der Entwicklungssysteme

Das Kapitel Konfiguration der Entwicklungssysteme befasst sich mit den Arbeiten, welche notwendig waren, um die im Kapitel 4 (Entwicklungssysteme) beschriebene Software korrekt verwenden zu können. In diesem Kapitel wird lediglich jene Software beschrieben, welche von einer üblichen, standardisierten Installation abweicht. Dies sind nachfolgend die Server und Eclipse als IDE für den Clientmod und die Serverplugins.

5.1 Windows Server 2012 R2

Die Software „Windows Server 2012 R2“ wurde zu Beginn der Diplomarbeit am Schulinternen Server unter den Projektservern als virtuelle Maschine unter dem Namen „minecraft.htl-perg.ac.at“ installiert. Der Zugang zum Server erfolgt über eine Verbindung auf Ares (den schulinternen Server) durch die Software VMware vSphere Client. Zum Login werden Matrikelnummer, Schulpasswort sowie die IP des Ares benötigt.

5.1.1 Die wichtigsten Installationen am Server:

- Aktuelle VMWare Tools
- Dropbox
- BungeeCord Bukkit Minecraft Server
- Google Chrome

5.1.2 Technische Informationen des Servers:

- Betriebssystem: Windows Server 2012 R2
- Hostname.(Domainname): minecraft.(htl-perg.ac.at)
- IP: 10.114.57.7 (Klasse C)
- Gateway: 10.114.57.254
- DNS1: 172.27.2.10
- DNS2: 172.27.1.1

5.2 BungeeCord Bukkit Minecraft Server

Um BungeeCord in Betrieb zu nehmen, müssen als Vorbereitungen die einzelnen Bukkit Server aufgesetzt werden. Dafür müssen für die drei Server, welche später gleichzeitig laufen installiert werden. Jeder Server hat dabei geringfügige Abweichungen von den anderen.

5.2.1 Installation der einzelnen Bukkit Server

1. Download der „spigot.jar“ – Datei von der Website www.getspigot.org
2. Erstellen von drei Verzeichnissen, in welchen sich später die einzelnen Server befinden und kopieren der „spigot.jar“ – Datei in jeden Ordner
3. Für jeden Ordner eine Batch – Datei erstellen, in welcher sich folgender Code befindet, um die „spigot.jar“ – Datei mit bestimmten Argumenten zu starten:

```
@echo off

java -Xms1024M -Xmx2048M -XX:MaxPermSize=128M -jar spigot_server.jar

pause
```

Abbildung 15: Batch-Code zum Starten eines Bukkit Servers

4. Nach einem ersten Starten der Server, wird eine Reihe von Downloads durchgeführt und die einzelnen Server bekommen die typische Dateistruktur.

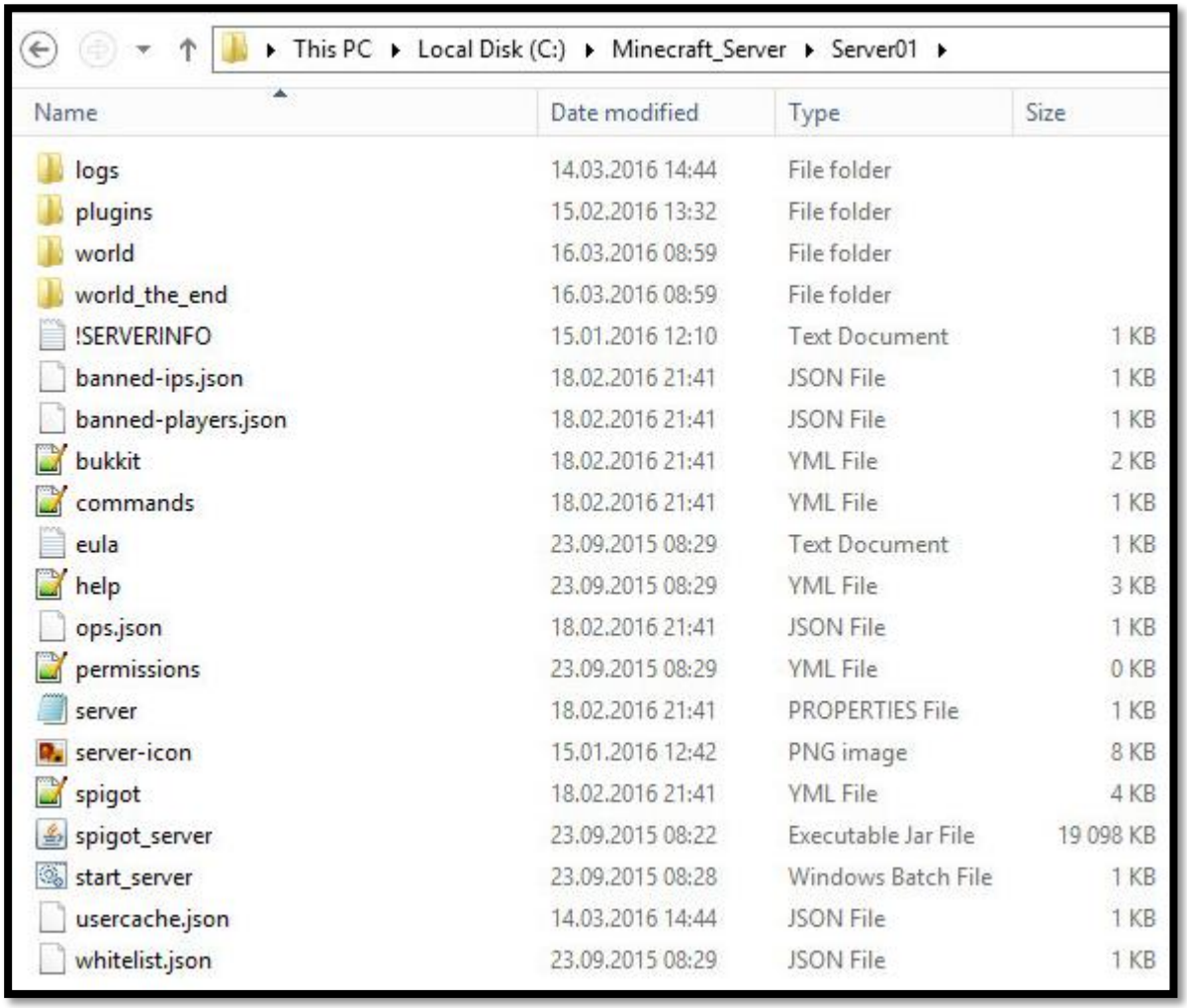


Abbildung 16: Dateistruktur Bukkit Server

5. Die wichtigsten Dateien hierbei sind folgende:
    - World: Der World – Ordner beinhaltet die Minecraft – Welt, die die einzelnen Spieler später betreten.
    - !Serverinfo.txt: Diese Datei wurde als Kurzinformation zum Server erstellt. Darin steht der Name des Servers, der Port, welche Plugins er benutzt und welche Funktion er hat.
    - Eula.txt: Die offizielle Endbenutzer Lizenzierung
    - Server.properties: Alle Einstellungsmöglichkeiten zum Server, diese Datei wird später noch genau erklärt.

5.2.2 Konfiguration der einzelnen Bukkit Server

Übersicht der Server

Port 25565 BungeeCord		
Port 25566:	**Port 25567**	**Port 25568**
Main-Lobby (Hub)	**Game-Lobby 1**	**ESG 1**

Unterschiede der Server

Wie in der Übersicht dargestellt, hat jeder Server eine eigene Funktionalität und einen eigenen Port. Die Funktionalitäten unterscheiden sich wie folgt:
In der Main-Lobby und der Game-Lobby sollen sich die Spieler nicht verletzen können, am ESG-Server jedoch schon. Deswegen müssen die „Gamemodes“ der Spieler standardmäßig geändert werden. Außerdem soll es in den Lobbys nicht möglich sein, Blöcke abzubauen. Im Spielmodus ESG ist dies jedoch essentiell. Auch sollen nur im ESG-Spielmodus aggressive Mobs spawnen. Abschließend dürfen im Gegensatz zur Main-Lobby auf die anderen beiden Server maximal 24 Spieler joinen, da diese Spieleranzahl in ähnlichen Spielmodi verwendet wird und dem Originalfilm „Tribute von Panem – The Hunger Games“ entnommen ist.

In Folge dieser Unterschiede wurden folgende Properties der eigenen Server geändert:

- Gamemode: Zum Einstellen, ob ein Spieler Blöcke abbauen darf
- Difficulty: Zum Einstellen, ob aggressive Mobs spawnen
- PvP: Zum Einstellen, ob sich Spieler gegenseitig verletzen dürfen
- Max-Players: Zum Einstellen der maximalen Spieleranzahl
- Server-Port: Der verwendete Port
- Server-IP: Die Server-IP unter der der Server erreichbar ist

5.2.3 Installation von BungeeCord

Die Installation von BungeeCord ist jener von den einzelnen Minecraft Servern ähnlich. Der neueste Build kann von Spigot gedownloaded werden und wird erneut durch ein Batch-Skript geöffnet. Nach dem Download des Builds wird also zuerst ein Verzeichnis bungee am Server erstellt und der Build und eine Batch-Datei mit folgendem Inhalt erstellt

```
java -jar BungeeCord.jar
PAUSE
```

Abbildung 17: Aufruf BungeeCord

Nach dem ersten Starten des Servers wird wieder eine Reihe von Ordnern und Dateien heruntergeladen. Danach kann der Server wieder heruntergefahren werden, da für BungeeCord noch eine Reihe von Konfigurationen notwendig sind.

Abbildung 18: Dateien des BungeeCord Servers

5.2.4 Konfiguration von BungeeCord

Um BungeeCord korrekt in Betrieb zu nehmen, ist eine Reihe von Konfigurationen in der „config.yml“ – Datei notwendig. Um die Anzahl der Spieler, welche sich auf dem gesamten Servernetzwerk befinden dürfen, zu regeln, kann eine Höchstgrenze bei „player_limit” festgelegt werden. Soll diese Anzahl unendlich sein, so wird „-1” eingetragen. Dies ist beim Mine4Glory – Server der Fall, da die Ressourcen des Servers momentan bei weitem nicht ausgeschöpft sind.

Zusätzlich können „Permissions“ für die einzelnen BungeeCord Befehle festgelegt werden. Am besten ändert man die Rechte aller möglichen Befehle so, dass diese lediglich von Administratoren benutzt werden dürfen.

Um BungeeCord die einzelnen Server, welche sich im Servernetzwerk befinden, zuzuweisen, sollten diese zuerst gestartet werden und der Port der einzelnen Server eingetragen werden. Zu jedem dieser Ports braucht man nun noch den eindeutigen Servernamen, unter dem BungeeCord den Server dann kennt. Außerdem kann man für die Server noch eine „modt“ angeben. Dies ist eine Kurzbeschreibung des Servernetzwerkes, welche angezeigt wird, wenn sich ein Spieler einen bestimmten, dem Servernetzwerk zugehörigen, Server unter dessen Port in seine Minecraft-Serverliste einspeichert.

```
servers:
  s01:
    motd: '&1Lobby of Mine4Glory'
    address: 10.114.57.7:25566
    restricted: false
  s02:
    motd: '&1Server 02'
    address: 10.114.57.7:25567
    restricted: false
```

Abbildung 19: BungeeCord Serverkonfiguration

Unter „modt“ kann man dann die Servernetzwerkkurzbeschreibung eingeben, welche angezeigt wird, wenn der Spieler den Server ohne Port in seine Minecraft-Serverliste einspeichert.

Unter „default-server“, „fallback-server“ und „force-default-server“ wird eingetragen, welcher Server die Hub darstellt, auf welchen Server weitergeleitet wird, wenn ein Fehler beim Verbinden auf einen anderen Server auftritt und ob der Standardserver bei jedem Join eines Spielers als Ziel ausgewählt wird.

Das Attribut „host“ bezeichnet die Server-IP, durch welche der Server zugänglich ist. Wie bereits in Kapitel 4.2 (BungeeCord Bukkit Minecraft Server) erwähnt, wird der Standardport durch BungeeCord belegt. Deswegen wird der IP des Servers noch „:25565“ zum Festlegen des Ports beigefügt.

Schlussendlich wurde noch ein 16x16 Pixel großes Serverbild eingefügt, welches den Spielern in der Minecraft-Serverliste angezeigt wird.

Abbildung 20: Der Server in der Minecraft-Serverliste

Das Ende der Konfigurationsarbeiten bildet die Erstellung einer Batch-Datei, welche nun das gesamte Servernetz hochfährt. Dabei werden zuerst die einzelnen Server hochgefahren und danach BungeeCord.

5.3 Eclipse

5.3.1 Java Development Kit (JDK)

Das JDK wird benötigt, um Java für den Gebrauch in der JRE zugänglich zu machen und um Java Komponenten kompilieren zu können. Das Kit wird zum Entwickeln und Debuggen von Java Programmen verwendet. Die neueste Version vom JDK steht auf der offiziellen Oracle Website zum Download bereit. Nach dem Download der in etwa 200MB großen Windows 64 oder Windows 86 Executable Datei wird das Java Installationsprogramm aufgerufen und der Benutzer kann sich zwischen einer vollständigen oder einer Teilinstallation entscheiden. Bei einer kompletten Installation der Datei wird auch die versionsgleiche JRE (welche im unteren Kapitel genauer beschrieben wird) mitinstalliert.

5.3.2 Java Runtime Environment (JRE)

Zum Ausführen von Java Programmen wird ein Java Runtime Environment benötigt. Während sich das JDK um die Benutzung und Verwaltung von vorgefertigten Klassen und Bibliotheken, zum Entwickeln und Debuggen von Java Programmen kümmert, bezieht die JRE ihre Arbeit mit der Ausführung dieser. Bei einer vollständigen Installation des JDK's wird die aktuelle JRE mitinstalliert. Um die Installation jedoch fertigzustellen und um das gesamte Betriebssystem mit der neuen Java Version vertraut zu machen, müssen noch Umgebungsvariablen von Windows gesetzt werden.

Um zu den Umgebungsvariablen zu gelangen und diese zu ändern, muss eine Reihe von Aktionen durchgeführt werden.

1. Öffnen der Systemsteuerung
2. Anzeige auf „kleine Symbole“ oder „große Symbole“ ändern
3. „System“ aufrufen
4. Mit Administratorrechten die erweiterten Systemeinstellungen öffnen
5. Den Button „Umgebungsvariablen“ drücken
6. In der Gruppe „Systemvariablen“ befindet sich eine „PATH“ – Variable, diese muss am Ende um das Verzeichnis der JDK und JRE Installation erweitert werden. Dazu wird der Pfad des „bin“ – Verzeichnisses jeweils der JDK und JRE Installation mit einem „;“ vorangestellt hinzugefügt.
7. Schlussendlich muss noch die „JAVA_HOME“ Variable gesetzt werden, dazu wird eine neue Variable mit diesem Namen angelegt und der JDK Installationspfad eingefügt

Wichtig: Nicht das „bin“ – Verzeichnis und ohne „;“ – Symbol

5.3.3 Eclipse für die Serverprogrammierung

Apache Maven

Apache Maven wird benötigt, um die entwickelten Java Programme für den Minecraft Server ausführbar zu kompilieren. Das bedeutet, es wandelt die Java Programme in die Plugins um. Maven übernimmt dabei eine zentrale Informationsverwaltung für den Build, die Protokolle und die Dokumentation.

Eclipse IDE for Java Developers

In Eclipse werden die Java Projekte, Packages und Klassen mit deren Methoden verwaltet und programmiert. Es ist eine vollständige Entwicklungsumgebung und unter den Minecraft Bukkit Plugin Developern die beliebteste IDE (Integrated Development Environment). Um die Plugins zu programmieren, muss dem Projekt noch eine Referenz zur aktuellen Bukkit.jar zugewiesen werden. Außerdem ist es hilfreich, die Bukkit Javadoc Location zu setzen, um eine Dokumentation für Klassen und Methoden in Eclipse integriert bereitgestellt zu haben.

Maven Integration Plugin für Eclipse (=M2Eclipse)

M2Eclipse stellt eine Integration für Apache Maven in die Entwicklungsumgebung Eclipse dar. Dabei stellt es folgende Features zur Verfügung:

- Starten von Maven Builds in Eclipse
- Builden von Programmen aufgrund den Einstellungen von Maven's pom.xml
- Automatisches downloaden von benötigten Dateien von Maven Repositories aus dem Web.
- Wizards zum Erstellen neuer Maven Projekte mit deren pom.xml, um aus einem normalen Java Projekt ein Maven Projekt zu machen
- Schnelle Suche für Dependencies, nach denen gebuilded wird in allen Maven Repositories aus dem Web

5.3.4 Eclipse für die Clientprogrammierung

Forge

Forge ist eine modding API, welche es erleichtert Mods zu erstellen und sicherzustellen, dass diverse Mods untereinander kompatibel sind. Es wird benötigt, wenn man clientbasierte Erweiterungen für Minecraft programmieren möchte. Forge ist kompatibel mit Eclipse und wird nach dem Download über das Kommandofenster installiert. Forge erstellt daraufhin ein Package, in welchem man Zugriff auf alle Minecraft-Basisklassen hat. In diesem Package kann man auch ein neues Mod-Package erstellen. In diesem wird der selbst erstellte Mod implementiert.

6 Funktionalitäten

Im Kapitel Funktionalitäten werden alle konkreten Erweiterungen des Spieles durch unsere Diplomarbeit angeführt. Dies sind einerseits die servererweiternden Plugins und andererseits die clienterweiternden Modifikationen.

6.1 Funktionalitäten der Serverplugins

Nachfolgend werden alle Serverplugins in ihrer Funktionsweise und der Auswirkung auf den Spielbetrieb detailliert beschrieben. Insgesamt wurden vier Plugins programmiert, ein Plugin je Minecraft Server und ein Plugin, welches auf allen Servern läuft. Außerdem wichtig ist ein besonderes Package der Plugins, welches alle Ausgaben an den einzelnen Client kontrolliert; das Message – Package.

6.1.1 Hub – Plugin

Das Hub-Plugin besteht aus der Main-Klasse, allen Listenern und dem Message-Package. Im Unterschied zu den beiden anderen Plugins, welche jeweils auf einem

Server installiert sind, sind in der Hub-Lobby keine Kommandos implementiert.

Main

In der Main-Klasse werden alle Listener registriert. Das bedeutet, dass man dem PluginManager des Plugins alle Listener übergibt.

```
PluginManager pm = Bukkit.getPluginManager();
pm.registerEvents(new PlayerListener(), this);
pm.registerEvents(new EntityListener(), this);
```

Abbildung 21: Registrieren von EventListener Klassen

Listener

Es gibt zwei verschiedene Listener Klassen. Eine für alle Entities und eine für Player. In jeder dieser Klassen gibt es dann die einzelnen EventHandler. Es ist wichtig, dass Methoden, welche bei einem Event gefeuert werden auch die Annotation @EventHandler besitzen. Zusätzlich zu dieser Annotation kann auch die Priority angegeben werden. Nachfolgend findet sich ein EventHander, welcher für alle Entites des Servers das EntityDamageEvent negiert. Dadurch kann kein Lebewesen durch Schaden verletzt werden.

```
public class EntityListener implements Listener
{
    @EventHandler(priority = EventPriority.HIGHEST)
    public void onEntityDamageEvent(EntityDamageEvent e)
    {
        {
            e.setCancelled(true);
        }
    }
}
```

Abbildung 22: Beispiel eines EntityEventHandlers

Um allen Spielern, welche sich momentan auf der Hub-Lobby befinden, keine unnötigen Chatausgaben zu liefern, wurden Events, welche eine Chatausgabe zur Folge haben negiert. Dazu zählen:

- PlayerAchievementAwardedEvent
- PlayerQuitEvent

Andere Events wurden jedoch den Umständen entsprechend verändert. So wird einem Spieler beim Joinen in die Lobby die Anzahl der Online-Spieler angezeigt und bei einem Versuch, in den Serverchat zu posten, wird die Nachricht durch das Message-Package umformatiert.

```
@EventHandler
public void onChatEvent(AsyncPlayerChatEvent e)
{
    PlayerToServerMessage.out(e.getPlayer().getDisplayName(), e.getMessage());
    e.setCancelled(true);
}
```

Abbildung 23: Originalnachricht negieren und formatierte Nachricht senden

6.1.2 Gamelobby – Plugin

Das Gamelobby – Plugin verwaltet alle Geschehnisse, während die Spieler, welche den Spielmodus ESG betreten wollen, warten.

Main

In der Main-Klasse werden zuerst die Kommandos, welche während des Aufenthalts auf dem Server angeboten werden, registriert.

```
ESGLobbyCommandExecutor ce = new ESGLobbyCommandExecutor(Main.instance);
getCommand("lobby").setExecutor(ce);
```

Abbildung 24: Registrieren eines Kommandos

Die Besonderheit an dieser Main-Klasse gegenüber den anderen ist jedoch ein Scheduler, welcher bei einer Änderung der Spieleranzahl reagiert, da der Sinn dieser Gamelobby ja das Warten auf mehrere Spieler ist, um dann den Spielmodus mit lediglich genügend Spielern zu starten. Diese Spielermindestgrenze kann nach Belieben festgelegt werden. Sobald diese überschritten ist, startet sich ein Scheduler, welcher für den einzelnen Spieler einen Countdown darstellt.

Durch diese Spielermindestgrenze ergeben sich natürlich verschiedene Möglichkeiten, was bei einem Spielerjoin oder Spielerleave passiert.

1. Es sind genügend Spieler online und der Countdown läuft noch nicht.
    a. Ausgabe, dass ein Spieler den Server betreten hat
    b. Countdown starten
2. Es sind genügend Spieler online und der Countdown läuft schon.
    a. Ausgabe, dass ein Spieler den Server betreten/verlassen hat
3. Es sind nicht mehr genügend Spieler online, jedoch läuft der Countdown.
    a. Ausgabe, dass ein Spieler den Server verlassen hat
    b. Den Countdown abbrechen
    c. Den Startwert zurücksetzen
4. Es sind nicht genügend Spieler online und der Countdown läuft noch nicht.
    a. Ausgabe, dass ein Spieler den Server betreten/verlassen hat
    b. Ausgabe, dass mehr Spieler gesucht werden

Listener

Wie auch beim Hub – Plugin, werden das EntityDamageEvent und alle störenden Chatevents abgebrochen. Zusätzlich wird jedoch beim PlayerJoinEvent und beim PlayerQuitEvent die Main-Klasse benachrichtigt, um die Spieleranzahl zu aktualisieren und um dann eine der vier oben erklärten Möglichkeiten durchzuführen.

Commands

Im Command-Executor befinden sich die Methoden, welche den Kommandos angehören.

/hub /lobby /l

Dieser Befehl schickt einen Spieler zurück zur Hub. Da diese Aktion jedoch wieder die Hilfe von BungeeCord benötigt, muss eine UTF-Nachricht an BungeeCord gesendet werden. BungeeCord bietet einen vorgefertigten Befehl für das Versenden von Spielern über Server. Dieser funktioniert durch das Senden des Schlüsselworts „connect“ und der Angabe des Servernamens, welcher in der BungeeCord-Config eingetragen ist.

```
ByteArrayOutputStream b = new ByteArrayOutputStream();
DataOutputStream out = new DataOutputStream(b);

try
{
    out.writeUTF("Connect");
    out.writeUTF("s01");
}
catch (IOException e)
{
    e.printStackTrace();
    PluginToPlayerMessage.out(MessagePrefix.HUB, "Befehl zurück zur Lobby hat nicht funktioniert"
}

PluginToPlayerMessage.out(MessagePrefix.HUB, "Connecting to Lobby", ((Player)sender));
p.sendPluginMessage(instance, "BungeeCord", b.toByteArray());
return true;
```

Abbildung 25: Versenden von Spielern via BungeeCord

/start

Dieser Befehl kann lediglich von Administratoren ausgeführt werden. Er setzt die restliche Wartedauer in der Lobby auf fünf Sekunden.

6.1.3 Extended Survival Games (ESG) – Plugin

Dieses Plugin ist das Herzstück der serverseitigen Diplomarbeit. Es repräsentiert den eigentliche Spielmodi, um den das gesamte Servernetzwerk aufgebaut ist. Folgende Funktionen wurden dabei umgesetzt:

- Spieler kämpfen gegeneinander ums Überleben
- Der letzte, überlebende Spieler gewinnt
- Eine sich verkleinernde Worldborder
- Random Teleport zu Beginn des Spiels um die Spieler zu verteilen
- Scoreboard – Anzeige um den Spielfortschritt zu zeigen
- Automatischer Teleport in die Border, wenn sich diese verkleinert
- Übliche Funktionen, wie auch bei den vorherigen Plugins

Gewinnerermittlung

Nach dem Tod eines Spielers wird dieser wieder zurück in die Lobby gesendet. Sobald nur mehr ein einziger Spieler im Spielmodus ESG ist, hat dieser das Spiel somit gewonnen.

Worldborder

Die Worldborder ist eine Grenze, welche die Spieler nicht verlassen dürfen. Zu Beginn des Spiels ist diese 2000x2000 Blöcke groß. Von den Koordinaten (-1000/-1000) bis zu den Koordinaten (1000/1000). Somit ist die Mitte bei (0/0). Im Lauf des Spiels, wenn die Zeit fortschreitet, wird die Border bedeutend kleiner, um ein rascheres Ende des Spielmodus zu erreichen, da sich die Spieler in kleineren Gebieten aufhalten und sich bekämpfen.

Random Teleport

Der zufällige Teleportationsvorgang zu Beginn des Spiels ermöglicht einen fairen, chancengleichen Beginn für alle Spieler. Außerdem verteilt er die Spieler, damit diese nicht direkt ohne ausreichend Equipment gegeneinander kämpfen müssen, sondern genügend Zeit haben, sich auf Kämpfe vorzubereiten.

Scoreboard

Das Scoreboard zeigt allen Spielern den momentanen Zeitfortschritt, die Anzahl der momentan lebenden Spieler und die Entfernung zur Border. Dabei sind diese Kennzahlen farbig markiert und das Scoreboard übersichtlich an der rechten Bildschirmseite dargestellt.

6.1.4 Message – Plugin

Da die BungeeCord Architektur aus mehreren einzelnen Minecraft-Servern besteht, kann eine Nachricht einen Spieler nicht erreichen, wäre er auf einem anderen Server als der Sender. Denn von Minecraft ist das Senden von Nachrichten nur innerhalb eines Servers möglich. Um Nachrichten jedoch Serverunabhängig an den richtigen Spieler zu senden, braucht man dieses Message – Plugin.

Darin wurde ein Kommandozeilenbefehl implementiert, welcher sich mit „/msg“ oder seinen Alias „/tell“ und „/t“ aufrufen lässt. Als Befehlszeilenargumente werden der Spielername jenes Spielers, welcher die Nachricht erhalten soll und die Nachricht selbst angehängt. Beim Abschicken dieses Befehls wird dann via eines Streams ein UTF-Code an BungeeCord geschickt, welches bereits eine gute API für solche Probleme bereitstellt. BungeeCord selbst sucht dann den Spieler auf allen Servern und übermittelt ihm seine Nachricht.

6.1.5 Das Message – Package

Das Message – Package ist für jegliche Chatausgabe zuständig. Um verschiedene Ausgaben von verschiedenen Teilen der Plugins (z.B. ESG, Lobby, Chat, Error, …) voneinander abzuheben, wurden dafür bestimmte Präfixe verwendet. Da es in Minecraft auch Farbcodes gibt, folgt ein kleiner Überblick über diese.

Sample	Code	Common Name	Foreground Color			Background Color		
			R	G	B	R	G	B
	&0	Black	0	0	0	0	0	0
	&1	Dark Blue	0	0	191	0	0	47
	&2	Dark Green	0	191	0	0	47	0
	&3	Dark Teal	0	191	191	0	47	47
	&4	Dark Red	191	0	0	47	0	0
	&5	Purple	191	0	191	47	0	47
	&6	Gold	191	191	0	47	47	0
	&7	Gray	191	191	191	47	47	47
	&8	Dark Gray	64	64	64	16	16	16
	&9	Blue	64	64	255	16	16	63
	&a	Bright Green	64	255	64	16	63	16
	&b	Teal	64	255	255	16	63	63
	&c	Red	255	64	64	63	16	16
	&d	Pink	255	64	255	63	16	63
	&e	Yellow	255	255	64	63	63	16
	&f	White	255	255	255	63	63	63

Abbildung 26: Minecraft Farbcodes

Die einzelnen Präfixe wurden aus designtechnischen Gründen wie folgt verwendet.

```
ESG = "§7[§6ESG§7]§r §a";
GAMELOBBY = "§7[§3Spiellobby§7]§r §a";
HUB = "§7[§5Hub§7]§r §a";
CHAT = "§7[Chat] [";
PRIVATECHAT = "§7[§ePrivat§7] [§e";
ERROR = "§7[§4Fehler§7]§r §4";
```

Abbildung 27: Präfixe für Chatnachrichten

Das Message – Package besitzt nun all diese Präfixe und vier verschiedene Methoden, welche eine Chatausgabe verursachen können. Dies sind eine globale Pluginnachricht, eine Pluginnachricht an einen Spieler, eine Nachricht eines Spielers an den gesamten Server oder eine Nachricht eines Spielers an einen andern Spieler.

Aus oben erwähnten Gründen wurden dann vier Methoden programmiert, welche eine erleichterte Ausgabe aus allen anderen Stellen der Plugins ermöglichen.

```
public final class PlayerToPlayerMessage
{
    public static void out(String sender, String message, Player reciever)
    {
        reciever.sendMessage(MessagePrefix.PRIVATECHAT + sender + "]" + message);
    }
}
```

Abbildung 28: Beispiel einer Ausgabe einer privaten Chatnachricht

6.2 Funktionalitäten des Client – Mods

Die Funktionalitäten der Clientmodifikation bestehen aus Erweiterungen in den Bereichen „Kreaturen“, „Items“ und „Blöcke“. Jede jeweilige Erweiterung besteht aus der Veränderung von Texturen, dem Modell und der Funktion. Bei Kreaturen ist die Funktion die jeweilige künstliche Intelligenz (KI).

6.2.1 Kreaturen

Nachdem die Kreaturen modelliert und designt wurden, müssen diese nun in das Programm implementiert werden. Die Implementierung erfolgt an verschiedenen Stellen, diese werden in der nächsten Tabelle beschrieben.

Implementierung	Zweck
Texturen	Die vorher designten Texturen müssen in einem bestimmten Package gespeichert werden, damit sie später vom Programm adaptiert werden können.
Modell	Hier werden wie vorher beschrieben die einzelnen vorher modellierten Boxen zu einer ganzen Kreatur zusammengesetzt. Zusätzlich werden auch beispielsweiße die Rotation des Kopfs oder der Füße implementiert.
KI (Künstliche Intelligenz)	Hier wird das Verhalten der Kreatur implementiert (z.B. Aggression, Geschwindigkeit, Stärke usw.). Es können verschiedene Attribute gesetzt werden, welche das Verhalten der Kreatur verändern.

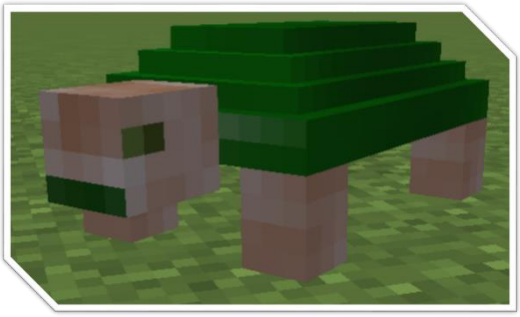
Bär

Implementierung	Zweck
Texturen	Die Texturen des Bären wurden so gewählt, dass sie an einen Braunbären erinnern.
Modell	Das Modell des Bären beinhaltet zusätzlich zu den Standardformen wie z.B. den Rumpf, den Kopf oder die Füße, noch zusätzlich eine Schnauze und einen Schwanz.
KI (Künstliche Intelligenz)  *Abbildung 29: Minecraft - Bär*	Ein Bär verhält sich immer aggressiv, das heißt er versucht den Spieler zu töten. Die KI des Bärs lässt nicht zu, dass dieser eine hohe Klippe herunterspringt oder sich durch andere Ursachen wie z.B. ein Feuer verletzt. Eine weitere Funktion der KI des Bären ist sein Verhalten gegenüber neutralen Tieren wie z.B. einem Schaf. Er versucht diese Tiere zu töten. Damit wurde versucht das natürliche Verhalten echter Bären nachzuahmen.

Wildschwein

Implementierung	Zweck
Texturen	Die Texturen des Wildschweins wurden so gewählt, dass sie an ein echtes Wildschwein erinnern. Eine Schwierigkeit dabei ist, die Stoßzähne zu designen, da diese sehr klein sind.
Modell	Das Modell des Wildschweins beinhaltet zusätzlich zu den Standardformen wie dem Rumpf, dem Kopf oder den Füßen noch zusätzlich eine Schnauze und Stoßzähne.
KI (Künstliche Intelligenz)  *Abbildung 30: Minecraft - Wildschwein*	Ein Wildschwein ist passiv-aggressiv. Das bedeutet, dass es sich so lange neutral verhält, bis es von einem Spieler angegriffen wird. Wenn ein Wildschwein angegriffen wird, verhält sich sofort das gesamte Rudel aggressiv und versucht den Spieler zu töten. Eine Schwierigkeit bei dem Entwickeln der KI war das Verhalten echter Wildschweine so gut es geht nachzuahmen. Dabei verloren die Wildschweine leider an Gefährlichkeit, da Wildschweine eher weglaufen als angreifen. Da der Grundgedanke aber ein anderer war, wurde für das Wildschwein eine aggressivere KI entwickelt.

Schildkröte

Implementierung	Zweck
Texturen	Die Texturen der Schildkröte wurden so gewählt, dass sie einer echten Wasserschildkröte ähneln. Eine Schwierigkeit dabei war, dass durch die geringe Größe nicht viel Platz für Details wie Augen oder Mund waren.
Modell	Das Modell der Schildkröte besteht aus vier Füßen, einem Kopf und einem Panzer. Der Panzer ist dabei aus vier Ebenen aufgebaut und jede dieser Ebenen musste implementiert und zusammengefügt werden.
KI (Künstliche Intelligenz)  *Abbildung 31: Minecraft - Schildkröte*	Die Schildkröte ist das einzige komplett friedliche Tier dieser Modifikation. Eine Schildkröte kann sich sowohl an Land als auch im Wasser fortbewegen. Dabei war eine Herausforderung, dass im Grunde zwei KI's geschrieben werden mussten. An Land bewegt sich die Schildkröte langsam voran, aber im Wasser ist sie ungefähr so schnell wie ein Spieler.

Swampy

Implementierung	Zweck
Texturen	Die Texturen des Swampys wurden so gewählt, dass sich eine möglichst giftige und gefährliche Erscheinung ergibt.
Modell	Das Modell des Swampys besteht aus zwei Füßen, zwei Händen und einem Kopf. Die Arme bewegen sich, wenn Swampy jemanden angreift.
KI (Künstliche Intelligenz)  *Abbildung 32: Minecraft - Swampy*	Swampy ist ein aggressives Monster, welches nur im Sumpf vorkommt. Wenn ein Spieler in Sichtweite ist, greift Swampy ihn an und vergiftet den Spieler durch seine Attacke. Die Geschwindigkeit der Kreatur ähnelt der eines Spielers, doch der Spieler kann der Kreatur davonlaufen, wenn er sprintet.

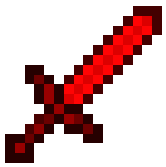
6.2.2 Items

Nach dem Designen der Items, müssen diese nun in das Programm implementiert werden. Die Implementierung erfolgt an verschiedenen Stellen, diese werden in der nachfolgenden Tabelle beschrieben.

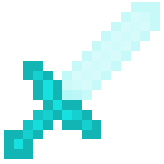
Implementierung	Zweck
Texturen	Die vorher designten Texturen müssen in einem bestimmten Package gespeichert werden, damit sie später vom Programm adaptiert werden können.
Modell	Hier wird die Anzeige des Items gespeichert. Diese vermittelt dem Programm die Proportionen des Items und wie es im Spiel dargestellt wird.
Funktion	Hier wird bestimmt, welche Funktion das Item hat. Es gibt verschiedene Elternklassen wie z.B. Essen, Schwert oder Rüstung. Diese Elternklassen bestimmen die Attribute der Kindklasse.

6.2.2.1 Waffen

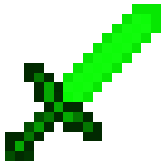
Feuerschwert

Implementierung	Zweck
Texturen	Die Klinge des Feuerschwerts ist hellrot mit einem dunkelroten Griff.
Modell	Das Modell des Feuerschwerts ist das Standardmodell eines Schwertes.
Funktion  *Abbildung 33: Minecraft - Feuerschwert*	Das Feuerschwert verursacht gleich viel Schaden wie ein Diamantschwert, zusätzlich verursacht es aber noch einen Feuereffekt, welcher den Gegner verbrennt.

Blitzschwert

Implementierung	Zweck
Texturen	Die Klinge des Blitzschwerts ist hellblau mit einem türkisenen Griff.
Modell	Das Modell des Blitzschwerts ist das Standardmodell eines Schwertes.
Funktion  *Abbildung 34: Minecraft - Blitzschwert*	Das Blitzschwert verursacht gleich viel Schaden wie ein Diamantschwert, zusätzlich schlägt ein Blitz ein, welcher den Gegner blendet und verletzt.

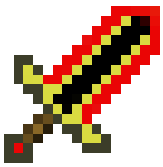
Giftschwert

Implementierung	Zweck
Texturen	Die Klinge des Giftschwerts ist hellgrün mit einem dunkelgrünen Griff.
Modell	Das Modell des Giftschwerts ist das Standardmodell eines Schwertes.
Funktion  *Abbildung 35: Minecraft - Giftschwert*	Das Giftschwert verursacht gleich viel Schaden wie ein Steinschwert, zusätzlich verursacht es aber noch einen Gifteffekt, welcher den Gegner vergiftet.

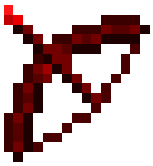
Spinnenschwert

Implementierung	Zweck
Texturen	Die Klinge des Spinnenschwerts ist hellgrau mit einem dunkelgrauen Griff.
Modell	Das Modell des Spinnenschwerts ist das Standardmodell eines Schwertes.
Funktion  *Abbildung 36: Minecraft - Spinnenschwert*	Das Spinnenschwert verursacht gleich viel Schaden wie ein Holzschwert, zusätzlich wird der Gegner verlangsamt und ihm damit die Flucht erschwert.

The Killer

Implementierung	Zweck
Texturen	Die Klinge des Schwertes ist schwarz mit einer gelb-roten Umrandung und mit einem braun-roten Griff.
Modell	Das Modell des Schwertes ist breiter als ein gewöhnliches Schwert.
Funktion  *Abbildung 37: Minecraft - The Killer*	Das Schwert „The Killer“ ist stärker als jedes andere Schwert im Spiel und vereint alle Effekte der vorher genannten Schwerter. Die Herstellung des Schwertes ist sehr schwer und benötigt zahlreiche Ressourcen, damit die Balance des Spiels beibehalten wird.

Feuerbogen

Implementierung	Zweck
Texturen	Der Bogen ist dunkelrot und die angezeigte Pfeilspitze ist hellrot.
Modell	Das Modell des Feuerbogens gleicht dem Standardmodell eines Bogens.
Funktion  *Abbildung 38: Minecraft - Feuerbogen*	Der Feuerbogen verursacht gleich viel Schaden wie ein normaler Bogen, zusätzlich verursacht der Feuerbogen aber noch einen Feuereffekt, welcher den Gegner verbrennt.

Doppelschussbogen

Implementierung	Zweck
Texturen	Der Doppelschussbogen hat die gleiche Farbe wie ein normaler Bogen.
Modell	Das Modell des Feuerbogens gleicht dem Standardmodell eines Bogens, jedoch sind zwei Pfeile in der Sehne eingespannt.
Funktion  *Abbildung 39: Minecraft - Doppelschussbogen*	Der Doppelschussbogen verursacht gleich viel Schaden wie ein normaler Bogen, jedoch schießt dieser Bogen zwei Pfeile auf einmal. Das bedeutet, dass theoretisch der doppelte Schaden mit einem Schluss erzielt werden kann

6.2.2.2 Rüstungen

Feuerrüstung

Implementierung	Zweck
Texturen	Die Feuerrüstung hat eine hellrote Farbe.
Modell	Das Modell der Feuerrüstung ist das Standardmodell einer Rüstung.
Funktion  *Abbildung 40 Minecraft - Feuerrüstung*	Die Feuerrüstung macht den Spieler, welcher die Rüstung trägt immun gegen Feuer und zusätzlich bietet die Rüstung so viel Verteidigung wie eine Diamantrüstung.

Blitzrüstung

Implementierung	Zweck
Texturen	Die Blitzrüstung hat eine hellblaue Farbe.
Modell	Das Modell der Blitzrüstung ist das Standardmodell einer Rüstung.
Funktion  *Abbildung 41: Minecraft - Blitzrüstung*	Die Blitzrüstung macht den Spieler, welcher die Rüstung trägt, immun gegen Blindheit und zusätzlich bietet die Rüstung so viel Verteidigung wie eine Diamantrüstung.

Schildkrötenrüstung

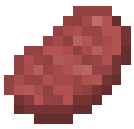
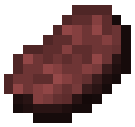
Implementierung	Zweck
Texturen	Die Schildkrötenrüstung hat eine braune Farbe.
Modell	Das Modell der Schildkrötenrüstung ist das Standardmodell einer Rüstung.
Funktion  *Abbildung 42: Minecraft - Schildkrötenrüstung*	Die Schildkrötenrüstung macht den Spieler, welcher die Rüstung trägt, langsamer, bietet dafür aber um einiges mehr Verteidigung als eine Diamantrüstung.

Giftrüstung

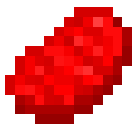
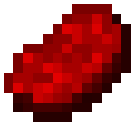
Implementierung	Zweck
Texturen	Die Giftrüstung hat eine hellgrüne Farbe.
Modell	Das Modell der Giftrüstung ist das Standard-modell einer Rüstung.
Funktion  *Abbildung 43: Minecraft - Giftrüstung*	Die Giftrüstung macht den Spieler, welcher die Rüstung trägt, immun gegen Vergiftung und zusätzlich bietet die Rüstung so viel Ver-teidigung wie eine Eisenrüstung.

6.2.2.3 Nahrung

Bärenfleisch

Implementierung	Zweck
Texturen	Das rohe Bärenfleisch hat eine hellrote Farbe und das gekochte eine dunkelrote.
Modell	Das Modell des Bärenfleisches ist an das Modell eines Schweinefleisches angelehnt.
Funktion  *Abbildung 44: Minecraft - rohes Bärenfleisch*  *Abbildung 45: Minecraft - gebratenes Bärenfleisch*	Das Bärenfleisch wird von einem Bären gedroppt, wenn er getötet wird. Das rohe Fleisch regeneriert ein Viertel des Hungerbalkens und das gekochte Fleisch regeneriert die Hälfte des Hungerbalkens.

Wildschweinfleisch

Implementierung	Zweck
Texturen	Das rohe Wildschweinfleisch hat eine hellrote Farbe und das gekochte eine etwas dunklere.
Modell	Das Modell des Wildschweinfleisches ist an das Modell eines Schweinefleisches angelehnt.
Funktion  *Abbildung 47: Minecraft - rohes Wildschweinfleisch*  *Abbildung 46: Minecraft - gebratenes Wildschweinfleisch*	Das Wildschweinfleisch wird von einem Wildschwein gedroppt, wenn es getötet wird. Das rohe Fleisch regeneriert ein Viertel des Hungerbalkens und das gekochte Fleisch regeneriert die Hälfte des Hungerbalkens.

Essensspieß

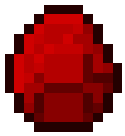
Implementierung	Zweck
Texturen	Der Essensspieß hat die Texturen von einem Rindfleisch, einer Karotte, einem Wildschweinfleisch und eines Holzstabes.
Modell	Das Modell des Essensspießes ist das eines normalen Items, eine Besonderheit bei diesem Item ist aber die Darstellung des Items wenn es gegessen wird.
Funktion  *Abbildung 48: Minecraft - voller Essenspieß*  *Abbildung 49: Minecraft - halber Essenspieß*  *Abbildung 50: Minecraft - leerer Essenspieß*	Der Essensspieß wird aus einem Rindfleisch, einer Karotte, einem Wildschweinfleisch und einem Holzstab hergestellt. Es regeneriert bei jeder Benutzung ein Drittel des Hungerbalkens und es kann viermal benutzt werden.

Diamantapfel

Implementierung	Zweck
Texturen	Der Diamantapfel ist hellblau und ähnelt der Farbe eines Diamanten.
Modell	Das Modell des Diamantapfels basiert auf dem eines Apfels.
Funktion  *Abbildung 51: Minecraft - Diamantapfel*	Der Diamantapfel wird aus einem Apfel umrundet von vier Diamanten hergestellt. Wenn der Apfel gegessen wird, regeneriert er ein Viertel des Hungerbalkens und erzeugt zusätzlich noch einen Regenerationseffekt und einen Verteidigungseffekt.

6.2.2.4 Erze

Feuerstein

Implementierung	Zweck
Texturen	Der Feuerstein ist von dunkelrot zu hellrot verlaufend.
Modell	Das Modell des Feuersteins basiert auf dem eines Diamanten.
Funktion  *Abbildung 52: Minecraft - Feuerstein*	Der Feuerstein wird aus einem Feuerblock hergestellt, indem dieser in einem Ofen geschmolzen wird. Der Feuerstein ist ein Item ohne Funktion und wird nur in Craftingrezepten benötigt.

Blitzstein

Implementierung	Zweck
Texturen	Der Blitzstein ist von dunkelgelb zu hellgelb verlaufend.
Modell	Das Modell des Blitzsteins basiert auf dem eines Diamanten.
Funktion  *Abbildung 53: Minecraft - Blitzstein*	Der Blitzstein wird aus einem Blitzblock hergestellt, indem dieser in einem Ofen geschmolzen wird. Der Blitzstein ist ein Item ohne Funktion und wird nur in Craftingrezepten benötigt.

6.2.2.5 Drops

Giftball

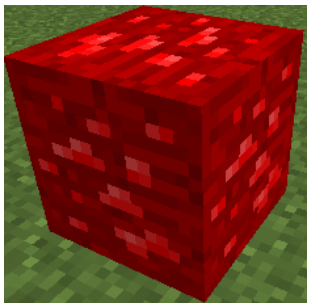
Implementierung	Zweck
Texturen	Der Giftball ist grün mit einer schwarzen Umrandung.
Modell	Das Modell des Giftballes basiert auf dem eines Schleimballes.
Funktion  *Abbildung 54: Minecraft - Giftball*	Der Giftball wird von Swampy gedroppt, wenn dieser getötet wird. Der Giftball ist ein Item ohne Funktion und wird nur in Craftingrezepten benötigt.

Schildkrötenpanzer

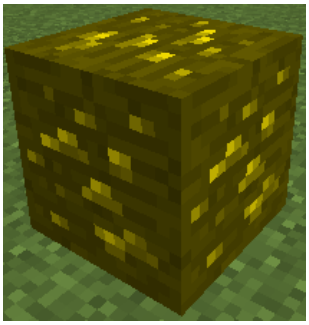
Implementierung	Zweck
Texturen	Der Schildkrötenpanzer ist dunkelbraun mit einer hellbraunen Umrandung.
Modell	Das Modell des Schildkrötenpanzers basiert auf keinem der Standard Minecraft-Modelle.
Funktion  *Abbildung 55: Minecraft - Schildkrötenpanzer*	Der Schildkrötenpanzer wird von einer Schildkröte gedroppt, wenn diese getötet wird. Der Schildkrötenpanzer ist ein Item ohne Funktion und wird nur in Craftingrezepten benötigt.

6.2.3 Blöcke

6.2.3.1 Feuerblock

Implementierung	Zweck
Texturen	Der Feuerblock ist dunkelrot mit hellroten Punkten.
Modell	Das Modell des Feuerblocks basiert auf dem Standardmodell eines Blockes in Minecraft.
Funktion  *Abbildung 56: Minecraft - Feuerblock*	Der Feuerblock kommt zufällig unter der Oberfläche in Höhlen vor, er hat eine Seltenheit von 15, das bedeutet er ist in 15 von 1000 Blöcken vorhanden. Wenn der Feuerblock in einem Ofen geschmolzen wird, erhält man einen Feuerstein.

6.2.3.2 Blitzblock

Implementierung	Zweck
Texturen	Der Blitzblock ist dunkelgelb mit hellgelben Punkten.
Modell	Das Modell des Blitzblocks basiert auf dem Standardmodell eines Blockes in Minecraft.
Funktion  *Abbildung 57: Minecraft - Blitzblock*	Der Blitzblock kommt zufällig unter der Oberfläche in Höhlen vor, er hat eine Seltenheit von 15, das bedeutet er ist in 15 von 1000 Blöcken vorhanden. Wenn der Blitzblock in einem Ofen geschmolzen wird, erhält man einen Blitzstein.

7 Probleme und Zukunftsaussichten

Bei der Entwicklung des Mods und des Plugins traten einige Probleme auf, manche konnten behoben werden, andere nicht. Im folgenden Abschnitt werden diese Probleme erörtert. Außerdem wird auf die Zukunft der Diplomarbeit eingegangen.

7.1 Zusammenführung der Komponenten

Ursprünglich war eine Zusammenführung der clientseitigen und der serverseitigen Erweiterung geplant. Durch die ständige Entwicklung des originalen Spieles wurde diese Zusammenführung jedoch sehr erschwert. Nachfolgend finden sich alle versuchten Lösungsansätze und momentan verfügbare API's, welche in Zukunft eine Zusammenführung ermöglichen könnten.

7.1.1 Cauldron

Cauldron ist ein Tool, welches es erlaubt, Forge Modifikationen und Craftbukkit Plugins auf einem Minecraft Server zu installieren.

Cauldron allows you to run Forge mods and CraftBukkit plugins on your Minecraft server.

[AKLIZ, 2016]

Wie man in dem Zitat sieht, sollte Cauldron das Zusammenfügen des Mods und des Plugins ermöglichen. Leider wurde Cauldron jedoch nie auf die Minecraft Version 1.8, mit der im Zuge der Diplomarbeit gearbeitet wurde, geupdated. Cauldron ist lediglich für Minecraft 1.7 verfügbar. Deshalb konnten wir dieses Tool nicht benutzen.

7.1.2 Sponge

Sponge ist ein weiteres Tool, welches es erlaubt, Forge Modifikationen und Plugins auf einem Minecraft Server zu installieren. Anders als bei dem Tool „Cauldron“, wurde Sponge bereits für die von uns unterstützte Minecraft Version 1.8 entwickelt. Jedoch unterstützt Sponge bisher nur Plugins, welche von den Sponge-Developern direkt entwickelt wurden und unterstützt keine momentan üblichen Bukkit-Plugins.

7.2 Server Befehlszeilencommands

Nachdem eine Datenbankanbindung für den Server via MySQL den zeitlichen Rahmen der Diplomarbeit gesprengt hätte, war es nicht möglich, alle Befehlszeilenkommandos, die am Diplomarbeitsbeginn als Projektziel festgelegt wurden, zu implementieren. Folgend finden sich Ansätze, welche bei einer zukünftig möglichen Datenbankanbindung, zu einer Lösung führen würden.

7.2.1 /friend

Beim Kommando /friend bräuchte man in der Datenbank die Zuordnung zweier Player via Assoziativtabelle. So hat ein Player mehrere andere, welche einem als Freund zugewiesen sind. Durch diesen Befehl wäre es auch möglich, Chatnachrichten nur an Freunde zu versenden, oder auch Partys nur für Freunde zugänglich zu machen.

7.2.2 /premium

Um Premium für Spieler möglich zu machen, müsste für jeden Spieler sein aktueller Rang in der Datenbank gespeichert werden. Dabei genügt für jeden Spieler eine Referenz via Fremdschlüssel, welcher auf den aktuellen Rang des Spielers zeigt. Mit diesem Befehl wäre es möglich, dass ranghöhere Spieler rangniedere Spieler aus den einzelnen Lobbys kicken und daher durch das Bezahlen von Geld den Vorteil bekommen, dass sie immer spielen können. Premium wäre dann monatlich oder lebenslang erhältlich.

7.2.3 /reply und /chatlog

Der Befehl /reply würde für den Benutzer die angestellte Nachricht an den letzten Spieler senden. Um dies zu ermöglichen, bräuchte man in der Datenbank zu jedem Spieler jeweils einen Spieler, der eben diese letzte Nachricht gesendet hat. In der Befehlsimplementierung müsste man dann jedoch noch abfragen, ob der betroffene Spieler auch online ist und ansonsten eine Fehlermeldung ausgeben. Beim Befehl /chatlog würden dann für jeden Spieler die letzten 10 Nachrichten gespeichert werden.

7.3 Spracheinstellungen am Plugin

Anders als beim Mod ist eine Spracheinstellung am Plugin nicht durch die Minecraft Engine vorgegeben. So müsste die spezifisch für jeden Spieler eingestellte Sprache in einer Datenbank gespeichert werden und jegliche Ausgabe an den Spieler an diese Sprache angeglichen werden. Um das Problem zu lösen, muss für jeden Spieler die Sprache in der Datenbank ausgelesen werden, die Strings dieser Sprache ausgelesen werden und dann vom Message-Package verwendet werden. Dadurch könnte jeder Spieler seine Sprache ändern.

7.4 Animation der Kreaturen

Ein großes Problem bei der Entwicklung der Modifikation war, dass Kreaturen oft ihre Gliedmaßen in unkontrolliertem Zustand rotierten. Dieses Problem entstand während der Exportierung des Modells von Techne in Java-Code. In diesen exportierten Koordinaten lag der Rotationspunkt einiger Blöcke nicht an dem zu erwartenden Punkt (wie im Techne Modell definiert), sondern wurde willkürlich gesetzt.

Dieser falsch gesetzte Rotationspunkt verursachte während der gesamten Entwicklung von Kreaturen ein wiederkehrendes, zeitlich aufwändiges Bugfixen, da solche Rotationspunkte mathematisch zu berechnen waren und dies eigentlich Aufgabe von Techne war. Somit konnte lediglich durch Berechnungen und Testungen festgestellt werden, an welcher Position der Rotationspunkt denn nun wirklich liegen muss, um eine ansprechende Animation zu erschaffen.

Literaturverzeichnis

AKLIZ. 2016. *Akliz.net*. [online]. Available from World Wide Web: <http://www.akliz.net/manage/knowledgebase/41/How-to-Install-a-Cauldron-Server.html>

BUKKIT. 2016. *Plugin Tutorial*. [online]. Available from World Wide Web: <http://wiki.bukkit.org/Plugin_Tutorial>

BUKKIT. 2016. *Setting Up The Workspace*. [online]. Available from World Wide Web: <http://wiki.bukkit.org/Setting_Up_Your_Workspace>

DROPBOX. *Dropbox*. [online]. Available from World Wide Web: <https://www.dropbox.com/>

ECLIPSE. *Eclipse*. [online]. Available from World Wide Web: <https://eclipse.org/>

ETERNALDOOM. *RealmsOfChaos*. [online]. Available from World Wide Web: <https://github.com/Eternaldoom/Realms-of-Chaos/tree/master/com/eternaldoom/realmsofchaos>

GAMEPEDIA. *Mods erstellen*. [online]. Available from World Wide Web: <http://minecraft-de.gamepedia.com/Mod/Mods_erstellen>

GIMP. *GIMP*. [online]. Available from World Wide Web: <https://www.gimp.org/>

JABELAR. *Creating Custom Entities*. [online]. Available from World Wide Web: <http://jabelarminecraft.blogspot.co.at/p/creating-custom-entities.html>

LONZBONZ. 2015. *Youtube - Lonzbonz*. [online]. Available from World Wide Web: <https://www.youtube.com/user/lonzbonz>

MIKASORBIT. *Youtube - Tutorials Modding*. [online]. Available from World Wide Web: <https://www.youtube.com/user/MikasOrbit>

MINECRAFT GAMEPEDIA. 2016. *Formatting Codes*. [online]. Available from World Wide Web: <http://minecraft.gamepedia.com/Formatting_codes>

MOJANG SYNERGIES AB. *Minecraft.net*. [online]. Available from World Wide Web: <https://minecraft.net/>

LITERATURVERZEICHNIS

PERG, HTL. 2015. *HTL Perg Diplomarbeiten*. [online]. Available from World Wide Web: <https://www.htl-perg.ac.at/kooperationen/diplomarbeit>

SPIGOTMC. 2016. *BungeeCord Configuration Guide*. [online]. Available from World Wide Web: <https://www.spigotmc.org/wiki/bungeecord-configuration-guide/>

SPIGOTMC. 2016. *BungeeCord Installation*. [online]. Available from World Wide Web: <https://www.spigotmc.org/wiki/bungeecord-installation/>

ZEUX. *Techne*. [online]. Available from World Wide Web: <http://techne.zeux.me/>

Abbildungsverzeichnis

ABBILDUNGSVERZEICHNIS