MUDs: Textorientierte Rollenspiele auf dem Computer

Jan Tobias Mühlberg

19. Februar 2003

While the philosophers and engineers sleep, the MUDers are at their computers, hour after hour, playing in the cyberspace. In Multi-User Dungeons [...] thousands of people share fantasy space, or „live“ electronically. They walk and talk, build and destroy, hug and have sex while sitting at isolated computer terminals scattered throughout the world.

[YOUNG 1994]

Inhaltsverzeichnis

1 Einführung

1983 schrieb William Gibson den Roman „Neuromancer“. Als einer der ersten Autoren brachte er die Vorstellung von einer „virtuellen Welt“, einer Welt, die nur als Simulation in einem Rechner existiert und die man nach belieben betreten und wieder verlassen kann, einer Welt, in der man sich relativ frei bewegen und mit anderen „Spielern“ interagieren kann, zu Papier. Der PC war zu diesem Zeitpunkt schon satte zwei Jahre alt und auch das Internet gab es bereits. Dennoch war alles in „Neuromancer“ ferne Zukunftsmusik. Kaum einer erachtete die beschriebene virtuelle Welt als etwas, was in absehbarer Zeit tatsächlich existieren könnte und niemand glaubte, daß man sie durch das anstöpseln einiger Elektroden an den Kopf des Spielers betreten könne - zumindest mit letzterem behielten all die Skeptiker damaliger Tage ja bis heute recht. Trotzdem wurden bereits damals heute so populäre Begriffe wie „Cyberspace“ oder „Cyberpunk“ geprägt. Vieles, was Gibson in seinem Buch schilderte fasziniert Autoren wie Leser von Science Fiction Literatur bis heute und die Ähnlichkeit von Gibsons dreckiger, düsterer Welt, in der skrupellose, drogensüchtige Söldner sich gegenseitig Straßenkämpfe liefern, mit vielen der in den vergangenen zwanzig Jahren entstandenen Computerspiele ist schlichtweg verblüffend. In der Tat wurde sein Buch von tausenden Menschen regelrecht verschlungen, die später versuchten seine Visionen, vermengt mit ihren eigenen, in Software umzusetzen und somit die ersten Rollenspiele für Computer entwickelten. Bis allerdings die ersten MUDs[1] entstanden sollte noch ein, für die heutige, schnellebige Informationsgesellschaft nicht unerheblicher Zeitraum vergehen.

2 Ein MUD? Was ist das?

Multi-User Dungeons im herkömmlichen Sinne sind Rollenspiele, in denen mehrere Spieler miteinander bzw. mit einer Reihe von NPCs[2] in einer virtuellen Realität interagieren können. Der Spieler nimmt hierbei seine virtuelle Umwelt über die in einem Computer verfügbaren Ausgabegeräte wahr und steuert das Verhalten seines Avatares. Ziel dessen ist üblicher Weise nicht ausschließlich die Lösung einer gestellten Aufgabe oder das Erreichen eines Spielzieles, vielmehr haben viele MUDs gar kein definiertes Spielziel oder Ende. Es handelt sich hierbei um sehr komplexe und sich ständig verändernde virtuelle Welten, die kaum mit konventionellen single-player Computerspielen

[1] MUD: Multi-User Dungeon

[2] NPC: Non Player Character, alles was in der virtuellen Welt „lebt“ und nicht Avatar eines Spielers ist

verglichen werden können. Curtis [CURTIS 1992] benennt drei Hauptfaktoren, die MUDs vom „Adventure- style computer game“ unterscheiden:

- A MUD is not goal-oriented; it has no beginning or end, no „score“, and no notion of „winning“ or „success“. In short, even though users of MUDs are commonly called players, a MUD isn't really a game at all.
- A MUD is extensible from within; a user can add new objects to the database [...] and [...] can describe whole new kinds of behavior for the objects they create.
- A MUD generally has more than one user connected at a time. All of the connected users are browsing and manipulating the same database and can encounter the new objects created by others. The multiple users on a MUD can communicate with each other in real time.

Prinzipiell unterscheiden sich die derzeit verfügbaren MUDs zum Einen durch die zur Visualisierung der virtuellen Realität genutzten Techniken, zum Anderen durch inhaltlichen Aspekte und das Spielkonzept. In Bezug auf die Technik kann man die MUDs in zwei große Gruppen aufteilen: Die Einen verwenden Graphik, die Anderen beschränken sich auf rein textuelle Beschreibungen von Szenerie und Handlungen. Diese Arbeit beschränkt sich, wie bereits aus dem Titel ersichtlich, vorrangig auf letztere Gruppe, wenngleich viele der hier beschriebenen Details auch ausnahmslos für graphische MUDs zutreffend sind. Wesentlich weiter lassen sich die vorhandenen Dungeons im Rahmen einer Analyse nach Inhalt und Spielkonzept auffächern, was jedoch nicht Gegenstand dieser Arbeit sein soll, nur soviel sei erwähnt: Das Spektrum reicht von historisch nahezu korrekt umgesetzten Begebenheiten über farbenfrohe Fantasy- und Science Fiction Geschichten bis hin zu virtuellen Nachbauten der wirklichen Welt[3]. Hinsichtlich des Spielkonzeptes sei bemerkt, daß in der Vergangenheit viele schöne Spiele, die die friedliche Interaktion von Spielern in den Vordergrund stellten, entstanden sind, wogegen es sich bei neuere Entwicklungen der Meinung des Autors nach vorrangig um „Hack and Slay“ - Spiele handelt.
MUDs nutzen meistens die Infrastruktur des Internets. Und als Teil dieses sind ihre Möglichkeiten und Grenzen stets von der Leistungsfähigkeit und Verfügbarkeit des Netzes abhängig, kann ihre Entwicklung fast nur in Zusammenhang mit der Entwicklung des Internets betrachtet werden. Der folgende Abschnitt soll die Meilensteine dieser Entwicklung darstellen.

[3]Ein schönes Beispiel hierfür ist das später noch vorgestellte *UNItopia* der Universität Stuttgart (siehe [UNITOPIA] bzw. *telnet://unitopia.de:3333*).

3 Die Geschichte der MUDs

Bereits Anfang der 70er Jahre wurde von Will Crowther am Palo Alto Research Centre der Xerox Corporation das erste Text-basierte Abenteuerspiel mit einer Tolkiens „Lord of the Rings“ entlehnten Fantasy-Welt programmiert. Donald Woods baute dieses Spiel im folgenden aus und gab ihm den Namen ADVENT. So wurde das Spiele-Genre der „Adventures“ begründet. Auch wenn man ADVENT bereits als eine virtuelle Realität im Sinne dieser Arbeit bezeichnen kann, fehlte ihm noch ein wichtiger Aspekt zu einem MUD: Das Spiel war nicht multi-player fähig. 1979 wurde ADVENT durch das von Alan Klietz vom Minesota Educational Computer Consortium entwickelte E*M*P*I*R*E[4] zu einem multi-player Adventure. Als erstes, von Anfang an mehr-Spieler fähiges Adventure gilt WIZARD, ebenfalls um 1970 programmiert.
Wenn man über die Geschichte der MUDs schreibt, darf man keinesfalls MUD1[5], ab 1978 von dem Studenten Ron Trubshaw an der University of Essex geschrieben, vergessen. Hierbei handelte es sich um ein in Anlehnung an das single-user-game „DUNGEON“ entwickelte und auf Donald Woods Code aufbauendes multi-player Spiel. In der Folgezeit entstanden viele weitere MUDs, die meisten dieser bauten auf MUD1 auf oder erweiterten es. Ihre größte Verbreitung hatte diese Spieleklasse zu Beginn der 90er Jahre, in dieser Zeit entstand auch das „legendäre“ LPMUD von Lars Pensjö (1989), das im folgenden zur bis heute meist genutzten Basis für neue MUDs wurde. Gleiches gilt für die ebenfalls von Pensjö entwickelte Programmiersprache LPC[6], auf die später noch näher eingegangen wird. Ebenfalls um diese Zeit öffnete auch UNItopia als ältestes MUD Deutschlands seine virtuellen Pforten, viele folgten ihm. Kurzzeitig besaß sogar die Fachhochschule Brandenburg ein eigenes, LPMUD bzw. UNItopia entlehntes MUD. Mangels studentischer Beteiligung innerhalb der FHB entschlossen wir uns jedoch Anfang 2001, die Arbeit daran einzustellen.
Um die Mitte der 90er Jahre tauchten dann, motiviert durch die steigenden Kapazitäten der Netze, wie auch deren zunehmende Verfügbarkeit, die graphischen MUDs auf. Hierzu gehören beispielsweise ULTIMA online und CrossFire[7].

[4] das spätere „Scepter of Goth“

[5] ursprünglich MUD, um Verwechslungen mit dem gleichnamigen Spiele-Genre vorzubeugen wurde es jedoch in MUD1 umgetauft

[6] LPC: Lars Pensjö C, eine von C und C++ abgeleitete Sprache zur Programmierung virtueller Realitäten

[7] siehe http://www.real-time.crossfire.com

4 Die Technik

Der folgende Abschnitt soll dem interessierten Leser einen kleinen Einblick in die bei der Implementation und Nutzung von MUDs angewandten Techniken geben.

4.1 Der MUD-Server

Das Herzstück eines jeden MUDs ist ein MUD-Server - ein Programm, daß auf einem Rechner irgendwo im Internet läuft und mit dem alle Spieler eine Verbindung aufbauen, dem sie ihre Handlungen übermitteln und das ihnen die Auswirkungen dieser Handlungen bzw. die auf ihren Avatar wirkenden Umwelteinflüsse mitteilt. Die Aufgaben eines solchen Servers sind dementsprechend vielfältig.
Jedoch stellt der Server nur ein relativ starres Gerüst bereit, häufig lediglich die nötigen Kommunikationsschnittstellen und einen Interpreter. Dieser wiederum interpretiert dann eine „Realitätsbeschreibungssprache“ wie beispielsweise LPC, in der dann das eigentliche MUD, also all die Räume, NPCs, Straßenbahnen und auch die Avatare der Spieler selbst, programmiert sind. All dies befindet sich in einer Bibliothek, der MUDLib, auf die im nächsten Abschnitt eingegangen wird.
Bei den textorientierten MUDs gibt es derzeit verschiedene Server, die gebräuchlichsten sind wohl LDMUD und LPMUD, wobei ersterer als ein Abkömmling des LPMUD gesehen werden kann.
Wie bereits erwähnt hält ein MUD-Server Verbindungen zu den einzelnen Clients offen, hinter jeden Client verbirgt sich üblicher Weise ein Spieler, der seinen Avatar steuert. Jedoch erlauben heutige Server auch etwas, was man als „Inter-MUD-Kommunikation“ bezeichnen könnte: Die Server sind in der Lage, untereinander Informationen auszutauschen. Das gibt einem Spieler die Möglichkeit, aus dem MUD, in dem er sich befindet, nach-zuschauen, ob evtl. ein anderer Spieler - Freunde und Bekannte - gerade in einem anderen MUD spielt. Er kann mit diesem anderen Spieler sogar mehr oder weniger in Echtzeit chatten oder ihm einen Brief schreiben. Da diese Funktionalität den Programmierern der virtuellen Realität voll zur Verfügung steht, wurde sie oft in so ansprechende wie realitätsnahe Dinge wie Fernmeldezentralen und Poststellen verpackt. Das Verhalten der dort „arbeitenden“ NPCs ist der Realität oft sogar viel zu ähnlich:

```
Du stehst im Verkaufsraum der oertlichen Poststelle. Rechts von
Dir stehen einige Regale mit diversen Verpackungsmaterialien.
Man benoetigt sie zum versenden von Briefen und Paketen. Auf
```

```
der linken Seite befinden sich zwei Telephonkabinen und genau
vor Dir befindet sich ein Postschalter. Hinter der Glasscheibe
scheint ein Postbeamter zu sitzen.
Ausgaenge: Sueden
> schaue postbeamter an
Der Postbeamte liegt mit dem Kopf auf seinem Tresen. Er scheint
zu schlafen. Vermutlich gibt es hier nicht viel zu tun.
> hoere
Du hoerst das gleichmeassige Roecheln des schlafenden
Postbeamten.
> klopfe gegen scheibe
Du klopfst gegen die Scheibe.
Der Postbeamte erwacht, reibt sich die Augen und sieht Dich
verschlafen an.
> ' ich wuerde gerne ein packet aufgeben
Du sagst: ich wuerde gerne ein packet aufgeben
Der Postbeamte mustert Dich.
Der Postbeamte sagt: Packet? Bist' wohl Informatiker, wie? Auf
Deutsch heissen die Dinger Pakete. Na gut, wo soll's denn
hingehen?
> ' Andreas@Virtu
Der Postbeamte legt sich seine Zeitung zurecht und schlaeft darauf
ein.
alFHB
Du sagst: Andreas@VirtualFHB
Der Postbeamte murmelt etwas.
> schaue Postbeamter an
Der Postbeamte liegt mit dem Kopf auf seinem Tresen. Er scheint zu
schlafen. An der Scheibe vor ihm haengt ein Schild.
> lies schild
+-----------------+
| MITTAGSPAUSE    |
| von 12 - 14 Uhr |
+-----------------+
> sueden
Du verlaesst das Postgebaeude.
Du stehst jetzt auf einem grossen Platz. Vor Dir befindet sich das
Postgebaeude. Du kannst Dich in alle Richtungen weiterbewegen.
> schaue himmel an
Du betrachtest den Himmel...
Deiner groben Schaetzung nach ist es jetzt etwa 11:45 Uhr.
```

4.2 Die MUDLib

Wie bereits im letzten Abschnitt angedeutet, ist die MUDLib jener Platz, an dem sich die eigentliche virtuelle Welt befindet. Das Wort kommt vom englischen Begriff *Library*, Bibliothek, und genau das ist es auch: Die MUDLib ist prinzipiell nicht mehr als ein (großer) Haufen Dateien auf einem Datenträger, auf die der MUD-Server zugreift und die alle Informationen über Räume, Objekte und Avatare enthalten. Anstelle von einzelnen Dateien bietet es sich auch an, diese Informationen konsistent in Datenbanken abzulegen, verschiedene MUD-Server unterstützen dies bereits.
MUDLibs gibt es wahrscheinlich fast so viele, wie es MUDs gibt. Auch wenn die meisten neuen MUDs auf die Bibliotheken der etablierten Dungeons aufbauen, gibt es doch in jedem einige Dinge, die grundlegend anders sind. Hierzu zählen beispielsweise Währungs- und Einheitensysteme, spezielle NPCs, Zeitrechnungen, Spielerinformationen, Beleuchtungssysteme und Ähnliches.
Die Implementation einer MUDLib ist ein nahezu monumentales Unternehmen. An ihr wird meist noch fleißig programmiert, wenn das MUD schon voller Spieler ist.

4.3 Die Clients

Als Client, also als jene Software, die der Spieler zur Kommunikation mit dem Server nutzt, eignet sich bei textorientierten MUDs prinzipiell jeder klassische TCP-Telnet Client. Für einen „echten“, hartgesottenen MUDer ist das auch die einzige Art und Weise, zu spielen: Der Spieler hat hier die Möglichkeit, seinen Avatar mittels per Tastatur einzugebender Kommandos zu steuern, beispielsweise so, wie es die beiden Beispiele in dieser Arbeit zeigen. Hierfür muß er natürlich die nötigen Kommandos kennen bzw. sie schnell in der online-Hilfe des MUDs nachschlagen.
Wem diese Methode nicht zusagt, dem steht es natürlich frei, einen MUD-Client, also ein Programm, das die Kommunikation zwischen Nutzer und Server dahingehend vereinfacht, als daß es die möglichen Kommandos über Buttons per Mausklick verfügbar macht, automatisch Karten des im MUD beschrittenen Weges anlegt - es ist beeindruckend, wie gut man sich auch als Programmierer in seinen eigenen Räumen verlaufen kann - und die Ausgaben des Servers etwas ansprechend aufbereitet. Häufig bieten solche Clients auch Möglichkeiten, Skripte für häufig durchgeführte Tätigkeiten zu schreiben. Es gilt jedoch als unhöflich und ist in den meisten etablierten MUDs auch verboten, diese Funktionalitäten beispielsweise zum regelmäßigen und vollautomatischen Abgrasen aller bekannter Schätze zu nutzen.
Für graphische MUDs werden im allgemeinen spezielle Clients benötigt.

4.4 LPC - Wie man virtuelle Realitäten beschreibt

LPC ist eine Programmiersprache. Sie wurde vor etwa 15 Jahren von Lars Pensjö entwickelt und eignet sich ganz hervorragend zur Implementation all der Räume, Rätsel, Monster und Avatare, die sich in MUDs üblicher Weise so finden. LPC ist eine objektorientiert, C bzw. C++ entlehnte Sprache.
Weil es nicht das Ziel dieser Arbeit ist, eine Programmiersprache zu lehren, wird dieser Abschnitt keinerlei theoretisches Grundwissen über MUDs und deren Konzeption und Programmierung vermitteln. Es soll lediglich ein winzig kleiner Einblick in die Art und Weise, wie man hier vorgeht und in die Möglichkeiten, die der Programmierer hat, gegeben werden. Aus diesem Grund wird dieser Abschnitt vorrangig den Quellcode eines Raumes der VirtualFHB wiedergeben und anhand dieses Beispiels dem interessierten Leser eine grobe Vorstellung von der Sprache LPC vermitteln. Darüber hinaus soll am Ende dieses Abschnittes gezeigt werden, wie sich der Raum dem Spieler darstellt.

Der Quelltext
Nun also zum versprochenen Quellcode eines MUD-Raumes. Dieser gliedert sich in diesem Falle auf mehrere Dateien auf, der Einfachheit halber sollen im Folgenden nur zwei davon block- bzw. zeilenweise erläutert werden.

eiche_oben.c
Diese Datei definiert den Raum selbst, sowie alle darin befindlichen Objekte:

```
1  inherit "/i/room";
```

Wie bereits weiter oben erwähnt ist LPC eine objektorientierte Sprache. Dementsprechend wird jedes neue Objekt von etwas bekanntem abgeleitet und erweitert. Man spricht in diesem Zusammenhang auch von Vererbung. Natürlich gibt es bereits ein Standardobjekt für Räume. Dieses wird hier als Ausgangspunkt genutzt.

```
2  #include <stats.h>
3  #include <config.h>
```

Die meisten Programmiersprachen gestatten das direkte Importieren von Quellcodedateien in eine neue Datei. Das ist auch in LPC möglich. An dieser Stelle werden zwei Dateien importiert, in denen einige im folgenden zu nutzenden Konstanten und Voreinstellungen definiert sind.

```
4  void reset();
```

Auch so etwas wie Forward-Declarations gibt es in LPC. Hier wird eine Funktion *reset()* deklariert. Reset hat keine Parameter und keinen Rückgabewert - wozu auch. *reset()* wird nämlich periodisch vom MUD-Server aufgerufen um den Raum zu „erneuern“. Dies ist ab und zu nötig, weil der Raum, nachdem ein Spieler ihn betreten und dort irgend etwas gemacht hat, nicht mehr in Ausgangsstellung ist und es dementsprechend anderen Spielern u.U. unmöglich wäre, die gleichen Tätigkeiten auszuführen. Darüber hinaus hinterlassen Spieler oft und überall irgendwelche Gegenstände - abgebrannte Fackeln, beschädigte Waffen und Ähnliches. Diese kosten den Server unnötig viel Speicher und sollten deswegen ab und an einmal entsorgt werden, was ebenfalls von der *reset()*-Funktion gemacht werden kann.
Die an dieser Stelle deklarierte Funktion macht erst einmal gar nichts, sie wird an anderer Stelle gefüllt.

```
 5  object skorpion;
```

Auch im Weiteren genutzte Objektbezeichner sollten man Anfang der Datei deklariert werden.

```
 6  void create()
```

Wird ein Raum erstmalig von einem Spieler betreten, läd der MUD-Server die entsprechende Quellcode-Datei und führt die Funktion *create()* aus, in der dann die Einzelheiten des Raumes definiert werden.

```
 7   {
 8    set_short("In der Eiche");
 9    set_long(wrap("Du stehst in einer relativ stabilen "
10                  "Astgabel der Eiche, es ist ziemlich "
11                  "Dunkel hier, da durch das dichte Laub "
12                  "kaum Licht zu dringen vermag. Links "
13                  "ueber Dir waechst eine Mistel."));
13    set_smell("Du riechst nichts besonderes.\n");
14    set_noise("Du hoerst das leise Rauschen der Blaetter "
15              "- zum Glueck ist es Windstill...\n"
16              "Gelegentlich piepsen rechts ueber Dir "
17              "einige Voegel.\n");
```

Ein jeder Raum hat einige grundsätzliche Eigenschaften. Hierzu gehören eine Kurzbeschreibung, die mit *set_short()* angegeben wird und eine etwas längere Beschreibung dessen, was der Spieler sieht. Letztere kann mittels *set_long()*

spezifiziert werden. Die Funktion *wrap()* kümmert sich nur um korrekte Zeilenumbrüche beim MUD-Client. Optionale Attribute eines Raumes sind beispielsweise der Umgebungsgeruch (*set_smell()*) und die Umgebungsgeräusche (*set_noise()*).

```
17   add_exit ("eiche_mitte.c","runter");
```

Ausgänge sind auch äußerst wichtig. Hier wird ein sichtbarer Ausgang definiert, der vom Spieler über das Kommando „runter“ genutzt werden kann. Tippt der Spieler also „runter“ ein, läd der MUD-Server die Datei „eiche_mitte.c“ und verfrachtet den Spieler dort hinein.
Unsichtbare Ausgänge gibt es auch, hier liegt es beim Spiele, das Entscheidende Kommando, also beispielsweise die Richtung in die er zu laufen hat, selbst erraten.

```
18   add_v_item(([
19    "name":"eiche",
20    "id":({"eiche", "Eiche", "baum", "Baum"}),
21    "gender":"weiblich",
22    "plural":0,
23    "feel"  :"Der alte Stamm hat Unmengen von Dellen "
24             "und Beulen.",
25    "long"  :"Eine riesige, maechtig gewaltige, "
26             "vermutlich sehr alte Eiche. Du kannst "
27             "allerdings nicht sehen, wie hoch sie "
28             "wirklich ist - ihre Blaetter versperren "
29             "Dir die Sicht. Allerdings haengst Du hier "
30             "schon knappe dreissig Meter ueber dem "
31             "Erdboden."]));
```

Hier wird ein sog. *v_item*, ein virtuelles Objekt deklariert. Virtuellen Objekte können aus Sicht des Spielers zwar betrachtet, befühlt und berochen werden, sie können jedoch nicht manipuliert, d.h. mitgenommen, angezündet, aufgegessen oder ähnliches, werden. Sie existieren lediglich um dem Programmierer etwas Arbeit und dem MUD-Server etwas Verwaltungsaufwand abzunehmen. Objekte, also auch virtuelle Objekte haben immer einen Namen und eine Menge von IDs, also Bezeichnern, über die ein Spieler auf sie zugreifen kann. Darüber hinaus ist ihnen grundsätzlich ein Geschlecht zugeordnet. Dies ist nötig, weil ein MUD-Server zur Kommunikation mit dem Spieler immer bestrebt ist, halbwegs natürliche Sätze zu bilden, in einer Sprache wie dem Deutschen geht das natürlich nur, wenn er dazu über ein gewisses Verständnis

für die deutsche Grammatik und natürlich auch über Informationen bezüglich der im einzelnen verwendeten Wörter verfügt. Auch *„plural“:x* ist hierfür wichtig - es spezifiziert, ob der gerade genutzte Begriff im Plural oder im Singular vorliegt.
Die beiden letzten Parameter definieren, was passiert, wenn der Spieler „fuehle Eiche“ bzw. „betrachte Eiche“ eingibt.

```
32  add_v_item(([
33   "name":"aeste",
34   "id":({"aeste","ast","astgabel","gabel","Aeste","Ast",
35          "Astgabel","Gabel"}),
36   "gender":"maennlich",
37   "plural":1,
38   "long":"Die Astgabel, in der Du gerade stehst, sieht "
39          "zwar stabil aus, so ganz sicher fuehlst Du "
40          "Dich dennoch nicht. Hoeher klettern kannst "
41          "Du allerdings nicht, weil Dir die Aeste dann "
42          "doch etwas zu duenn werden."]));

43  add_v_item(([
44   "name":"blaetter",
45   "id":({"blaetter","blatt","Blaetter","Blatt"}),
46   "gender":"weiblich",
47   "plural":1,
48   "long"  :"Du siehst viele, viele typische "
49            "Eichenblaetter. Zwischen ihnen haengen "
50            "gelegentlich ein paar Eicheln. Links ueber "
51            "Dir sind noch einige Mistelblaetter."]));

52  add_v_item(([
53   "name":"eicheln",
54   "id":({"eichel","eicheln","Eichel","Eicheln"}),
55   "gender":"weiblich",
56   "plural":1,
57   "long":"Jetzt erkennst Du, dasz es sich bei diesem "
58          "Baum um eine Stieleiche handelt."]));

59  add_v_item(([
60   "name":"mistel",
61   "id":({"mistel","mistelblaetter","Mistel",
62          "Mistelblaetter"}),
```

```
63    "gender":"weiblich",
64    "plural":0,
65    "long"  :"Die paar kleinen Mistelblaetter, die Du "
66             "von unten gesehen hast, entpuppen sich "
67             "bei naeherer Betrachtung als ziemlich "
68             "grosze Pflanze, die aus einem dicken Ast "
69             "der Eiche sprieszt. Irgendetwas Glaenzendes "
70             "scheint hier eingewachsen zu sein."]));

71   add_v_item(([
72    "name":"glaenzende",
73    "id":({"glaenzendes","glaenzen", "Glaenzendes",
74           "Glaenzen"}),
75    "gender":"saechlich",
76    "plural":0,
77    "long"  :"Da ist etwas in die Mistel eingewachsen. "
78             "Es glaenzt in dem spaerlichen Licht "
79             "golden, koennte ein Messer oder aehnliches "
80             "sein. Vielleicht kannst Du es herausziehen?"
81    ]));
```

Virtuelle Objekte gibt es in den meisten Räumen zu Hauf. Alles, was der Spieler üblicher Weise nicht konkret manipulieren muß, wird als virtuelles Objekt realisiert und weil MUDs ja realitätsnah sein sollen, d.h. weil der Spieler sich ja nach Möglichkeit alles, was er in der Raumbeschreibung findet auch explizit anschauen können soll, finden sich häufig wesentlich mehr virtuelle als tatsächliche Objekte.

```
82    reset();
83   }
```

Am Ende der *create()*-Funktion rufen wir gleich noch einmal die, immer noch nicht vollständig spezifizierte Funktion *reset()* auf um den Raum zu initialisieren.

```
84  void reset()
85   {
86    if (!present("skorpion",this_object()))
87     {
88      skorpion =
89       clone_object("/d/Vergangenheit/obj/skorpion.c");
```

```
 90      skorpion->move(this_object());
 91     }
 92   }
```

Die mysteriöse Funktion *reset()* - hier wird sie endlich definiert: *reset()* guckt jedes Mal, wenn sie aufgerufen wird nach, ob ein Objekt *skorpion*, das hatten wir in Zeile 5 deklariert, noch existiert. Solle es nicht da sein, beispielsweise weil es ein Spieler mitgenommen oder umgebracht hat, wird es neu erstellt. Der Skorpion ist ein NPC, ein Monster. Es wird von einem vordefinierten Objekt in der Datei */d/Vergangenheit/obj/skorpion.c* kopiert und anschließend mittels *move()* in den Raum bewegt.
Den Quellcode des Skorpions werden wir uns im folgenden Abschnitt ansehen.

```
 93  void init()
 94   {
 95    add_action("ziehe", "ziehe");
 96   }
```

init() ist eine weitere Funktion, die vom MUD-Server automatisch gestartet wird und zur Initialisierung des Raumes dient. In diesem Fall definiert *init()* mittels *add_action()* eine neue, vom Spieler über den Befehl „ziehe <irgendwas>“ nutzbare Funktion *ziehe()*.

```
 97  int ziehe(string s1)
 98   {
 99    object sichel;

100    if(!s1) s1="0";
101    if(lower_case(s1)=="glaenzendes" ||
          lower_case(s1)=="glaenzen")
102     {
103      if((!present("skorpion",this_object())) ||
104         (present("sichel",this_object())) ||
105         (present("sichel",this_player())))
106       {
107        write(wrap("Du ziehst an der Mistel herum, "
108                   "bekommst aber nichts richtig "
109                   "zu fassen - scheinbar hast Du Dich "
110                   "getaeuscht, da scheint nichts zu "
111                   "liegen."));
```

```
112        say(wrap(Der(this_player())+" zieht und und "
113                 "wuehlt in der Mistel herum, weiss aber "
114                 "scheinbar selbst nicht so recht, was er "
115                 "da sucht."));
116       }
117       else
118       {
119        write(wrap("Du ziehst eine alte, goldene Sichel "
120                   "aus der Mistel. Sie rutscht Dir "
121                   "jedoch aus den Haenden und bleibt in "
122                   "einer Astgabel haengen."));
123        say(wrap(Der(this_player())+" zieht einen "
124                 "Gegenstand aus der Mistel. Er bleibt "
125                 "in einer Astgabel haengen."));

126        sichel = clone_object("/obj/nahkampf_waffe");
127        sichel->set_name("sichel");
128        sichel->set_gender("weiblich");
129        sichel->set_id( ({"sichel","gegenstand"}) );
130        sichel->set_adjektiv("golden");
131        sichel->set_long("Ein kleine, goldene Sichel. "
132                         "Sie ist sehr scharf.\n");
133        sichel->set_weight(1);
134        sichel->set_damage(2, 15);
135        sichel->set_used_stats( ({STAT_STR, STAT_STR,
136                STAT_DEX}) );
137        sichel->set_skill_path( ({"skill", "offensiv",
138                "scharf", "sichel"}) );
139        sichel->set_learning_type(LEARNING_1);
140        sichel->set_life(100);
141        sichel->set_broken_message("Deine goldene "
142                "Sichel is voellig verbogen!\n");
143        sichel->set_value(100, 1000);
144        sichel->move(this_object());

145        skorpion->exec_command("toete", this_player());
146       }
147      return 1;
148     }
149     else return 0;
150   }
```

Dieser lange Block Code definiert die Funktion *ziehe()*. Nach der Deklaration eines neuen Objektes *sichel* wird überprüft, was der Spieler eigentlich eingegeben hat. Sollte das „ziehe Glaenzen“ oder „ziehe Glaenzendes“ gewesen sein, geht die Verarbeitung weiter: Es wird überprüft, ob der Skorpion noch da ist, ob bereits irgendwo eine Sichel rumliegt oder ob der Spieler sie vielleicht sogar schon hat. In Abhängigkeit vom Ergebnis dieser Vergleiche wird dann jeweils ein entsprechender Text ausgegeben. Dies geschieht jeweils zweimal - mit *write()* ausgegebene Nachrichten erreichen lediglich den Spieler, der die Aktion ausgeführt hat, im Gegensatz dazu gehen mittels *say()* erzeugte Nachrichten an alle anderen, ebenfalls in diesem Raum befindliche Spieler. Außerdem wird, falls die Bedingungen erfüllt sind, ein neues Objekt, nämlich die Sichel erzeugt und in den Raum bewegt (Zeilen 126 bis 144) und der Skorpion dazu gebracht, den betroffenen Spieler anzugreifen (Zeile 145), er scheint also so etwas wie ein Wächter zu sein.
Die Sichel ist eine Nahkampfwaffe, dementsprechend kann sie von einem Standardobjekt „nahkampf_waffe“ geklont und mit einigen neuen Eigenschaften versehen. Diese sind größtenteils selbsterklärend oder äquivalent mit den bei der Erzeugung des Raumes oder virtuellen Objekten genutzten. Interessant sind vielleicht noch die Funktionen *set_used_stats()* und *set_skill_path()*. Sie definieren für Objekte, welche Fähigkeiten bei Nutzung den Objektes gebraucht werden und in welchen Fähigkeiten ein Avatar Erfahrung bekommt, wenn er den Gegenstand nutzt. In diesem Fall sind das vor allem Kampffähigkeiten.

skorpion.c

Diese Datei spezifiziert einen NPC, nämlich den in Zeile 89 der Datei *eiche_oben.c* verwendeten Skorpion:

```
1  inherit "/i/monster/monster";
```

Standardobjekte gibt es für fast alles, u.A. auch für NPCs. Zugegeben, der Name „Monster“ ist etwas unglücklich gewählt, in diesem Fall hier jedoch treffend.

```
2  #include <stats.h>
3  #include <config.h>
```

Diesen Teil hatten wir bereits erläutert...

```
4  object stachel;
```

Der Skorpion ist natürlich bewaffnet. Mit einem Stachel. Wenn der Skorpion dann einmal tot ist, soll dieser Stachel auch liegenbleiben und dem Spieler zur anderweitigen Verwendung zur Verfügung stehen.

```
 5  void reset()
 6   {
 7    if (!present("stachel",this_object()))
 8     {
 9      stachel = clone_object("/obj/nahkampf_waffe");
10      stachel->set_name("skorpionsstachel");
11      stachel->set_gender("maennlich");
12      stachel->set_id( ({"stachel","skorpionsstachel"}) );
13      stachel->set_adjektiv("giftig");
14      stachel->set_long("Ein kleiner, gefaehrlicher "
15                        "Skorpionsstachel\n");
16      stachel->set_weight(1);
17      stachel->set_damage(3, 7);
18      stachel->set_used_stats( ({STAT_STR, STAT_STR, "
19               "STAT_DEX}) );
20      stachel->set_skill_path( ({"skill", "offensiv",
21               "scharf", "messer"}) );
22      stachel->set_learning_type(LEARNING_1);
23      stachel->set_life(50);
24      stachel->set_broken_message("Dein Skorpionsstachel "
25               "ist kaput.\n");
26      stachel->set_value(10, 40);
27      stachel->move(this_object());
28      stachel->set_hidden_until_next_move();
29     }

30    if (!present("seele",this_object()))
31     {
32      seele=clone_object("/obj/soul");
33      seele->move(this_object());
34     }

35    this_object()->exec_command("fuehre",stachel);
36   }
```

Der Stachel wird in der Funktion *reset()* erzeugt, damit der Skorpion regelmäßig einen neuen, intakten Stachel übergeholfen bekommt - Nahkampfwaffen nutzen sich nämlich im Kampf auch ab und gehen irgendwann einmal

kaput.
Neben dem Stachel bekommt der Skorpion auch noch eine Seele. Die Seele ist ein sehr spezielles Objekt. Sie ist nicht sichtbar und enthält all die Funktionalität, die ein Objekt, ganz gleich ob NPC oder Avatar braucht um zu sprechen, Emotionen auszudrücken, kurz um sich wie ein „Spieler“ zu benehmen und zur Interaktion fähig zu sein.

```
37  void create()
38   {
39    monster::create();

40    initialize("skorpion", 100);
41    set_koerpergroesse(1);
42    set_koerperform("vierbeiner");
43    give_hp(60);
44    set_id(({"skorpion"}));
45    set_gender("maennlich");
46    set_name("Skorpion");
47    set_short("Ein kleiner Skorpion");
48    set_long("Ein kleiner, dunkelbrauner Skorpion, der "
49             "sehr boese und giftig aussieht.");
50    load_chat(10,({"!echo $Der() krabbelt zwischen den "
51                    "Blattern der Mistel umher.",
52                   "!echo $Der() putzt seine Kauwerkzeuge.",
53                   "!echo $Der() puhlt sich mit seinem "
54                    "rechten Hinterbein irgendwelche "
55                    "Nahrungsreste aus seinen Scheren."}));
56    set_weight(3);
57    set_aggressive(0);

58    reset();
59   }
```

Hier wird das eigentliche Monster initialisiert. Die Eigenschaften sind wiederum selbsterklärend. Der Skorpion[8] ist darüber hinaus in der Lage, einige Aktivitäten (Zeilen 50 - 55) scheinbar auszuführen, tatsächlich bekommen anwesende Spieler natürlich nur einen Text angezeigt, u.A., damit sie wissen, daß der Skorpion in der Mistel sitz - man sollte Spieler eben nicht ohne

[8]dem Autor ist natürlich bewußt, daß Skorpione wesentlich mehr als vier Beine haben, jedoch muß man beim Programmieren ab und an vereinfachen - die prinzipiell unendliche Variationsvielfalt der Natur zu implementieren ist eh unmöglich.

Vorwarnung in ihr Verderben laufen lassen. Wichtig ist auch Zeile 57: Der Skorpion ist von sichaus nicht aggressiv. Er wird erst durch die Aktion eines Spielers etwas angriffslustig.
Bequem könnte man noch ettliche Seiten mit Ausführungen über LPC und das Design von von MUD-Räumen vollschreiben, Ziel dieses Aufsatzes ist das jedoch nicht. Es sollte ein grober Überblick gegeben werden und das ist hiermit getan. Der Interessierte Leser wird unter [UNITOPIA] bzw. unter [MUD-DE] genügend Informationen finden.

Der fertige Raum
Nachdem nun obiger Quelltext einen Raum mit einem versteckten „Schatz“, der Sichel, und einem NPC definiert und damit die Geduld des an technischen Details nicht interessierten Lesers schon arg strapaziert haben dürfte, jetzt wieder ein kleiner praktischer Exkurs: Wie sieht der Raum aus und was kann man darin machen.

```
Du stehst in einer relativ stabilen Astgabel der Eiche, es
ist ziemlich Dunkel hier, da durch das dichte Laub kaum
Licht zu dringen vermag. Links ueber Dir waechst eine Mistel.
Die Sonne geht auf.
      Weiter: Runter.
Ein kleiner Skorpion.
> Der Skorpion putzt seine Kauwerkzeuge.
betrachte blaetter
Du siehst viele, viele typische Eichenblaetter. Zwischen ihnen
haengen gelegentlich ein paar Eicheln. Links ueber Dir sind
noch einige Mistelblaetter.
> betrachte mistel
Die paar kleinen Mistelblaetter, die Du von unten gesehen
hast, entpuppen sich bei naeherer Betrachtung als ziemlich
grosze Pflanze, die aus einem dicken Ast der Eiche sprieszt.
Irgendetwas Glaenzendes scheint hier eingewachsen zu sein.
> Der Skorpion krabbelt zwischen den Blattern der Mistel umher.
betrachte skorpion
Ein kleiner, dunkelbrauner Skorpion, der sehr boese und giftig
aussieht.
Er ist topfit.
> betrachte glaenzendes
Da ist etwas in die Mistel eingewachsen. Es glaenzt in dem
spaerlichen Licht golden, koennte ein Messer oder aehnliches
sein. Vielleicht kannst Du es herausziehen?
```

```
> Der Skorpion krabbelt zwischen den Blattern der Mistel umher.
ziehe messer heraus
Wie bitte?
> ziehe glaenzendes heraus
Du ziehst eine alte, goldene Sichel aus der Mistel. Sie rutscht
Dir jedoch aus den Haenden und bleibt in einer Astgabel haengen.
Der Skorpion greift Dich mit seinem Skorpionsstachel an.
> Der Skorpion trifft dich mit seinem Skorpionsstachel hart.
Der Skorpion trifft dich mit seinem Skorpionsstachel hart.
zaubere blitz skorpion
Du sendest einen Blitz vom Himmel.
Ein lautes Donnern ist zu hoeren.
Du hast den Skorpion getoetet.
> betrachte
Du stehst in einer relativ stabilen Astgabel der Eiche, es
ist ziemlich Dunkel hier, da durch das dichte Laub kaum
Licht zu dringen vermag. Links ueber Dir waechst eine Mistel.
Die Sonne geht auf.
      Weiter: Runter.
Die Leiche eines Skorpions.
Eine goldene Sichel.
> betrachte leiche
Das ist die Leiche eines Skorpions.
Sie scheinen schwer zu sein.
        Sie enthaelt:
Ein giftiger Skorpionsstachel.
> ...
```

5 Wie man ein MUD eröffnet

Nachdem man als Spieler einige Zeit in verschiedenen MUDs verbracht hat, möchte man evtl. selbst ein MUD eröffnen oder zumindest eines aktiv mitgestalten. In den meisten Fällen ist letzteres die sinnvollere Methode, der Zeitaufwand, den man in die Neueröffnung eines solchen Projektes investieren muß, ist gewaltig und, was noch schlimmer ist, am Anfang kaum abschätzbar. Aus eigener Erfahrung einer MUD-Pleite kann ich sagen, daß vieles, was ganz vielversprechend anfängt, von einigen enthusiastischen Programmierern getragen wird, sehr schnell zu einer riesigen, abstoßenden und für alle Beteiligten durchweg frustrierenden Baustelle werden kann.
Doch prinzipiell steht es natürlich jedem Frei, sein eigenes MUD zu starten. Hierfür braucht man vorallem ersteinmal ein gute Spielidee, etwas was das eigene MUD von all den Großen und Etablierten unterscheidet und was den Spieler letztlich dazu animiert, sich im neuen MUD blicken zu lassen. Wenn man denn glaubt, ein Konzept, das einem zumindest einen gewissen „Kundenstamm“ sichern könnte, gefunden zu haben, kann man sich mit dessen Planung befassen. Dafür benötigt man einige tatkräftige Mitstreiter und vorallem viel Zeit. Die Planungsphase sollte unter Anderem auch die Auswahl der zu nutzenden MUDLib umfassen: Fast alle etablierten MUDs bieten ihre Libs im Internet an. Unbedingt sollte man sich alle wenigstens Ansatzweise anschauen, bevor man eine Entscheidung fällt. Einen Rechner, auf dem das ganze laufen soll, der dazu über eine gute Internetanbindung verfügt, braucht man natürlich auch noch. Und dann kann es eigentlich schon losgehen. Einen MUD-Server kann man sich beispielsweise bei [MUD-DE] herunterladen, ihn zu installieren sollte nach dem Studium der mitgelieferten Dokumentation kein Problem sein. Ebenso verhält es sich mit der MUDLib, für die man sich entschieden hat.
Häufig beginnt jetzt eine Phase, in der alle Beteiligten wild und oft ziemlich planlos drauflos programmieren. Irgendwann wird man dann feststellen, daß man von dem, was man eigentlich machen wollte, noch genau so weit entfernt ist wie ganz am Anfang - und dann setzt die große Frustration ein. Offensichtlich sollte man, wenn man ein eigenes MUD eröffnen möchte, sich vorher intensiv Gedanken darüber machen, wieviel Zeit man zu investieren bereit ist. Sicherheitshalber kann man dabei gleich planen, was passiert, wenn das „Bauvorhaben“ plötzlich anfängt, das Doppelte oder Dreifache der eingeplanten Zeit und Arbeitskraft zu verschlingen.
Darüber hinaus nützt einem das beste Spielkonzept nichts, wenn man seine Mitstreiter nicht bei Laune halten und dazu motivieren kann, das Projekt fortzusetzen. Eine einzelne Person hat kaum eine Chance in absehbarer Zeit ein fesselndes und spielbares MUD zu programmieren. Ein frisch-

gebackener MUD-Admin sollte dementsprechend etwas mit Menschen umgehen können. Vorallem muß es in einer Gruppe von Entwicklern feste Regeln bezüglich der Zuständigkeiten im einzelnen bzw. bezüglich der Entscheidungsfindung im Falle globaler Fragen geben. Häufig beginnen Spieler ihre eigenen MUDs zu schreiben, weil ihre Ideen und Vorstellungen an andere Stelle nicht oder zu wenig beachtet wurden, weil ihnen die Entscheidungsfindung in den großen MUDs zu undemokrtisch war. Gerade für sie bedeutet es eine herbe Enttäschung, zu sehen, daß die eigene kleine Neuentwicklung noch viel undemokratische ist.
In diesem Sinne: Die Eröffnung eines MUDs will gut und sorgfältig geplant sein, viel Glück all jenen, die es probieren möchten.

Damit ist das Ende dieser kleinen Einführung über MUDs erreicht. Leider sind diese in letzter Zeit etwas aus der Mode gekommen und auch die Spieler scheinen nicht mehr zu werden, was jedoch niemanden daran hindern sollte, sich trotzdem mit der Materie zu befassen bzw. seiner Phantasie etwas freien Lauf zu lassen...

6 Quellenverzeichnis

Ein Großteil der Informationen für die ersten zwei Abschnitte dieser Arbeit stammen aus Lars Hitzings hervorragender Diplomarbeit mit dem Titel „Escape - Identität im Cyberspace“, verfügbar unter <http://www.mud.de/Forschung/ESCAPE.pdf>. Darüber hinaus wurden folgende literarische Quellen und Web-Dokumente zur Erstellung dieser Ausarbeitung verwendet:

[CURTIS1992]	CURTIS, Pavel: „Mudding: Social Phenomena in Text-Based Virtual Realities.“, Intertrek, 3/1992, S. 26-34
[MUD-DE]	Web-Dokument: http://www.mud.de
[UNITOPIA]	Web-Dokument: http://unitopia.uni-stuttgart.de
[UTZ96]	UTZ, Sonja: Kommunikationsstrukturen und Persönlichkeitsaspekte bei MUD-Nutzern, verfügbar im Internet unter http://www.tu-chemnitz.de/phil/psych/professuren/sozpsy/Mitarbeiter/Utz/Diplom1.htm
[UTZ99]	UTZ, Sonja: Soziale Identifikation mit virtuellen Gemeinschaften - Bedingungen und Konsequenzen, 1999, Pabst-Verlag, Chemnitz
[YOUNG1994]	YOUNG, Jeffrey R.: „Textuality in Cyberspace. MUDs and Written Experience.“ 1994