Inhaltsverzeichnis

1 Vorwort

In diesem Tutorium soll die Erstellung eines einfachen Charakters, der in die Gruppe aufgenommen kann, erklärt werden. Im folgenden werde ich einen solchen Charakter als NPC bezeichnen. Die gesamte Erweiterung des Spieles wird als Mod bezeichnet. Meine Kenntnis beruht auf meinen beschränkten Erfahrungen und ist somit nicht unbegrenzt. Ein deutschsprachiges Forum, das sich mit der Thematik befasst findet man unter www.rosenranken.net . Ich werde die Erstellung eines NPC mit Hilfe der von mir benutzten Programme beschreiben. Über die Funktion von anderen Programmen kann ich keine Aussagen treffen. Von mir benutzt werden:

- WeiDU von Wesley Weimer: www.cs.berkely.edu/ weimer/bgate/#weidu
- Shadowkeeper von Aaron O'Neil: www.mud-master.com
- NearInfinity von Jon Olav Hauglid: www.idi.ntnu./ joh/ni/

Zur Konvention: das "*" steht hier für eine beliebige Zeichenkette (z.B. für einen Dateinamen). Der Syntax und spezielle Begriffe werden *kursiv* geschrieben.
Charaktere, die nicht in die Gruppe aufgenommen werden können auf dieselbe Weise erstellt werden. Auf die Unterschiede wird im folgenden hingewiesen.
Unbedingt ist darauf zu achten, dass die erstellten Dateien die DOS-Dateinamen-Konvention beachten (Länge der Dateinamen nicht länger als 8 Buchstaben!).

1.1 Abkürzungen

Hier einige Abkürzungen die ich verwende und die vielleicht nicht jeder kennt:

- HC = Hauptcharakter ist der Charakter der von Spieler generiert wurde, der spielinterne Name ist *Player1*
- DV = DeathVariable (Todesvariable) ist eine Variable, die den NPC gegenüber dem Spiel identifiziert; also so etwas wie der spielinterne Name; wird in der char.cre-Datei zugeordnet (siehe **??**)
- Mod = Modul/Modifikation; eine Abkürzung für das, was hier erklärt wird

1.2 Inhalt

Ich werde zuerst die Erstellung der *.cre-Datei erklären, dann in die Erschaffung der Dialoge einführen. Als nächster Schritt werde ich auf die benötigten Skripte eingehen. Zum Schluss werde ich erläutern, wie die einzelnen Teile zusammengefügt werden.

1.3 Änderungen der Version

1.3.1 zur 1_1

- Beschreibung der charJ.dlg
- Korrektur der Funktion der Bchar.dlg
- Einige Ergänzungen der Skripterläuterungen

1.3.2 zur 1_0

- zusätzliche Änderung der *.cre-Datei

1.3.3 zur beta-Version

- Einiges zur Datei Bchar.dlg
- Eingabe der *Death Variable*
- mehr interne Referenzen
- Bug im Charakterskript korrigiert.
- Ich versuche, auf die Anführungszeichen zu verzichten und benutze stattdessen die *kursive* Schrift. Dies soll klar machen, wann im Sysntax Anführungszeichen benützt werden müssen.

2 Erstellung der *.cre-Datei

2.1 Modifizierung der *.chr-Datei

Man erstellt den gewünschten Charakter im Multiplayer-Modus und exportiert ihn. Dabei braucht man nicht auf die Charakterpunkte (die Werte für Weisheit, Stärke, etc.) zu achten. Auch nicht wichtig ist die Verteilung der Waffenfertigskeitspunkte, Zauber, und Diebesfähigkeiten. Diese so erstellte *.chr-Datei öffnet man mit dem Shadowkeeper. Dort kann man als erstes die Charakterpunkte verändern (unter *Abilities*). Bei der Verteilung der Charakterpunkte sollte man darauf achten, dass man die Mindestanforderungen für eine Klasse/Rasse nicht unterschreitet. Die Mindestanforderung wird bei der Charaktererstellung im Spiel angezeigt. Ausserdem sollte man nicht zu viele Charakterpunkte verteilen, mehr als 90 sind unangebracht und führen leicht zu einem übermächtigen Charakter, der die Spielbalance stört. Aus demselben Grund sollte man nicht zuviele Eigenschaften auf 18 (oder sogar höher) setzen. Mein Vorschlag wäre so zwischen 70 und 85 Charakterpunkten und nur eine Eigenschaft auf 18 (19 bei solchen Charakteren, bei denen das bei der Charaktergenerierung möglich ist).
Unter dem nächsten Menüpunkt ist noch etwas zu ändern und zwar *Pc2* in *neutral* oder *anything* zu ändern.
Danach kann man noch die Waffenfertigkeiten neu vergeben. Dies geschieht unter dem Menüpunkt *Proficiencies* (eventuell bis zu dem Punkt scrollen) mit Hilfe der +/- Felder. Normalerweise ist die erste Spalte zu nutzen, bei einem umgelernten Charakter auch die Zweite. Hierbei sollte man einem Charakter nur die Fertigkeiten zuteilen, die er auch bei der Charaktererstellung erlangen könnte. Darüber hinausgehende Fertigkeiten sollten wohlüberlegt und gut begründet sein.
Danach kann man bei magiebegabten Charakteren deren Zaubersprüche austauschen (bei Bedarf). Dies geschieht unter dem Menüpunkt *Wizard* für Zauberer/Hexenmeister bzw. *Priest* für Kleriker/Druiden. Mit dem Feld *Add Spell* wird ein Zauber aus der Datenbank hinzugefügt (Feld *Assign to*). Mit *Mem+/Mem-* wird die Anzahl der memorierten Zauber bestimmt.
Die Diebeseigenschaften werden unter *Thieves* eingetragen. Unter *Abilities* können noch die Trefferpunkte verändert werden. Die angezeigten Trefferpunkte sind die erwürfelten (ohne Konstitutionsbonus).
Desweiteren kann man auch unter *Saving Throws* die Rettungswürfe verändert werden.
Normalerweise sollte man bei den Diebeseigenschaften nur umverteilen und die Rettungswürfe belassen.
Wenn man die Trefferpunkte verändern will, sollte man sich überlegen, warum. WENN man sie verändert sollte man darauf achten, nicht mehr als die maximal (minimal) möglichen Trefferpunkte zu verteilen (Maximalwürfelzahl beim Stufenanstieg multipllziert mit Stufe).
Priesterzauber sollte man nicht verändern. Man kann allerdings andere Spells memorieren; dabei ist jedoch darauf zu achten, die Anzahl der memorierten Zauber pro Stufe zu belassen.
Beim Modifizieren der Magierzauber belässt man entweder die Anzahl der Zauber pro Stufe oder man gönnt dem Magier noch ein paar zusätzliche Sprüche, die er von Schriftrollen gelernt hat. Allerdings muss man darauf achten, einem MagierSPEZIALISTEN nicht Zauber aus einer ihm nicht zugänglichen Schule zuzuweisen. Einem Hexenmeister sollte man keine zusätzlichen Sprüche

zuteilen, eine Veränderung ist gestattet. Wie auch bei den Priestersprüchen kann man den Magier andere Sprüche memorieren lassen, wobei die Anzahl der memorierten Sprüche pro Stufe erhalten werden muss. Will man die Anzahl der memorierten Zauber erhöhen, macht man dies unter dem Menüpunkt *Memorization* mit den Feldern *Max+/Max-*.

2.2 Eine höhere Stufe für den Charakter

Es gibt zwei Möglichkeiten einen Charakter auf eine höhere Stufe zu bringen.

1. ihm in seinem Skript Erfahrungspunkte zuzuweisen (siehe **??**)
2. seine *.chr-Datei verändern

Um die Werte in seiner *.chr-Datei zu verändern ist die Kenntnis der Eigenschaften pro Stufe aus AD&D erforderlich (für Hexenmeister, Barbar und Mönch aus D&D). Man muss die Terfferpunkte, Erfahrungspunkte (unter *Abilities*), Rettungswürfe und eventuell Waffenfertigkeiten verändern. Magier und Priester benötigen neue Zauber und mehr memorierte Zauber, Diebe (und Mönche) zusätzliche Diebesfertigkeiten. Bei NPCs sollte man im allgemeinen Methode 1 wählen. Will man, dass der Charakter sich in eine spezielle Richtung entwickelt, entscheidet man sich für Methode 2. Für einen Charakter, der nicht für die Aufnahme in eine Gruppe gedacht ist, kommt nur Methode 2 in Frage. Sollte jemand einen hochstufigen Charakter benötigen, der kein NPC sein soll, und besitzt nicht das AD&D Spielerhandbuch und kann das Handbuch nicht lesen, dann gibt es noch die verzweifelte Chance, einen NPC-Charakter zu erschaffen, ihm mittels eines Skripts die benötigten Erfahrungspunkte zukommen zu lassen (siehe **??**), den Stufenanstieg wie gewünscht durchführen, danach den NPC exportieren und die *.chr-Datei wieder in eine *.cre-Datei umwandeln.

2.3 Die Erstellung der *.cre-Datei

Die modifizierte *.chr-Datei wird nun unter einem anderen Namen gespeichert (wichtig, da sonst die Veränderungen ignoriert werden). Danach wird sie unter dem Menüpunkt *Tools* in eine *.cre-Datei konvertiert und als solche abgespeichert. Damit ist dieser Schritt abgeschlossen.

3 Die Dialoge

3.1 Was für Dialogdateien werden benötigt

Jeder Charakter, der auf sprachlicher Ebene mit der Gruppe interagieren, muss eine Hauptdialogdatei haben, die ich als char.dlg bezeichnen werde. NPCs brauchen zusätzlich noch eine Datei für den Rauswurf aus einer Gruppe, die ich charP.dlg nennen werde. Weiterhin sinnvoll ist eine Dialogdatei für Gespräche mit Gruppenmitgliedern, die ich als Bchar.dlg bezeichnen werde. Für die Reaktion eines NPCs auf bestimmte Ereignisse ist eine charJ.dlg nützlich. Soll der NPC auch in der Erweiterung “Thron des Bhaal” (TdB) Verwendung finden, müssen seine Dateien dafür ausgelegt sein. Aus Gründen der Übersicht und Kompatibilität empfehle ich das Anlegen von gesonderten Dateien, die als char25P.dlg, char25J.dlg und Bchar25.dlg bezeichnet werden.

3.2 WeiDU

WeiDU ist ein Programm mit dem die *.dlg Dateien elegant aus *.d Dateien generiert und in die dialog.tlk/dialogF.tlk integriert werden. Weiterhin ermöglicht WeiDU mit Hilfe einer *.tp2-Datei die reversible Installation des Mods. Auf die *.tp2-Datei wird im letzten Abschnitt eingegangen. Man kann auch mit Hilfe von NearInfinity die *.dlg-Dateien direkt editieren. Allerdings stellen die *.d-Dateien einen besseren Überblick über die Funktionsweise der Dialoge dar. Im Spiel vorhandene *.dlg-Dateien als Anschauungsmatrial zu benutzen wird empfohlen; diese können mittels WeiDU in die komfortableren *.d-Dateien umgewandelt werden.

3.2.1 Die Umwandlung *.d ↔ *.dlg mit WeiDU

Man entpackt die heruntergeladenen WeiDU-Dateien in ein beliebiges Verzeichnis, das im weiteren als *WeiDU-Verzeichnis* bezeichnet wird. Eine *.d-Datei, die in eine *.dlg-Datei übersetzt werden soll, muss in das *WeiDU-Verzeichnis* kopiert werden. Das Verzeichnis in dem BG2 installiert worden ist und in dem sich die Dateien *chitin.key, dialog.tlk, dialogF.tlk* befinden ist das *BG2-Verzeichnis*.
Um eine *.d-Datei aus einer *.dlg Datei des Spiels zu generieren, muss man die *MS-DOS Eingabeaufforderung* öffnen und sich in das *WeiDU-Verzeichnis* begeben. Dort gibt man dann folgenden Befehl ein:

*weidu --game BG2-Verzeichnis *.dlg*

Danach sollte die *.d-Datei im *WeiDU-Verzeichnis* zu finden sein.

Beispiel mit *E:\bg2* als *BG2-Verzeichnis* und *BKorgan.dlg* als gewünschter Datei:

weidu --game E:\bg2 BKorgan.dlg

Derselbe Befehl wandetl auch *.dlg-Dateien in *.d-Dateien um:

*weidu --game BG2-Verzeichnis *.d*

Beispiel mit *E:\bg2* als *BG2-Verzeichnis* und *Bvanim.d* als gewünschter Datei:

weidu --game E:\bg2 Bvanim.d

3.2.2 Der WeiDU-Syntax einer *.d-Datei

Der Syntax von WeiDU besteht aus Bedingungen, die abgefragt werden, und den Folgen:

IF ∼Bedingung∼

legt die *Bedingung* fest, die erfüllt sein muss, damit ein Dialog gestartet wird.

THEN BEGIN Dialogname SAY ∼Zeichensatz∼

legt den *Zeichensatz* fest der bei erfüllter Bedingung vom Charakter "gesprochen" (ausgegeben) wird. *BEGIN* erklärt dies zum Anfang des Gesprächs *Dialogname*.

Das Gespräch muss immer in der char.dlg des Charakters stehen, der das Gespräch beginnt.

Weiter geht es mit:

IF ∼Bedingung∼ THEN REPLY ∼Zeichensatz∼

wobei die *Bedingung* erfüllt sein muss, damit dem HC (HauptCharakter, der Spieler) die Antwort möglich ist. Nach einem *IF ... SAY* können mehrere Antwortoptionen (*IF ... THEN REPLY ...*) folgen. Man sollte auf jeden Fall darauf achten, dass mindestens eine Antwort immer möglich ist. Dies kann man dadurch erreichen, dass entweder eine Antwort keine Bedingung hat (*IF ∼∼ THEN REPLY ...*) oder dass zwei Antworten gegensätzliche Bedingungen haben.

Die Antwort kann noch zusätzlich erweitert werden:

IF ∼Bedingung∼ THEN REPLY ∼Zeichensatz∼ DO ∼Aktion∼

verknüpft eine *Aktion* mit dieser Antwort. Aktionen können vielfältig sein; die benötigten Aktionen werden an der entsprechenden Stelle erklärt. Eine häufige Aktion ist das Setzen einer Variablen.

Man kann auch auf eine gesprochene Antwort (den *~Zeichensatz~*) verzichten, dann besteht die Antwort in einer *Aktion* (nonverbale Kommunikation ;)):

IF ~Bedinung~ THEN DO ~Aktion~

Natürlich muss nach der Auswahl einer Antwort, das Gespräch entweder beendet oder weitergefürt werden. Beendet wird das Gespräch durch ein angefügtes *EXIT*, also:

IF ~Bedingung~ THEN REPLY ~Zeichensatz~ DO ~Aktion~ EXIT

Im Spiel erscheint dann das Feld *Dialog beenden*, das vom Spieler angeklickt werden muss.

Soll das Gespräch weitergeführt werden ist ein *GOTO nächsterDialogname* anzufügen:

IF ~Bedingung~ THEN REPLY ~Zeichensatz~ DO ~Aktion~ GOTO nächsterDialogname

Der nächste Dialog ist dann mit *IF ~Bedingung~ THEN BEGIN nächsterDialogname* zu starten. Eine einfache Methode Dialoge zu bennen, ohne dass es zu Doppelbenennungen, die zu vermeiden sind, kommt, ist, die Dialoge innerhalb einer Datei einfach durchzunummerieren (1,2,3,...). Die Nummerierung wird generell empfohlen, da sie für erweiterte Dialogoptionen benötigt wird.
Ein Dialog muss dann noch zum Schluss mit *END* geschlossen werden, so dass man eine *BEGIN ... END* Konstruktion hat.
Damit es nicht zu verwirrend wird ein Beispiel:

IF ~Bedingung~ THEN BEGIN 1
SAY ~Hallo!~
IF ~~ THEN REPLY ~Hallo!~ EXIT
IF ~Bedingung2~ THEN REPLY ~Wer seid Ihr denn?~ GOTO 2
IF ~Bedingung3~ THEN REPLY ~Hallo Idiot!~ DO ~Aktion~ GOTO 3
END

Noch zum Schluss: eine *.d-Datei muss mit BEGIN dlgname begonnen werden (z.B. *BEGIN char* liefert die char.dlg). Diesem *BEGIN* ist kein *END* zugeordnet. Sollen aus einer *.d-Datei mehrere *.dlg-Dateien generiert werden, schreibt man einfach ein weiteres *BEGIN nächstedlg* und dann den Syntax für die nächste *.dlg-Datei. Der Übersicht halber sollte man aber verschiedene Dateien verwenden. Zumindest sollte eine *.d-Datei nur die Dialoge eines Charakters enthalten.

3.2.3 Die char.dlg

In dieser Datei wird vor allem die Aufnahme in die Gruppe beschrieben. Wenn der Charakter kein NPC sein soll, stehen hier die Dialoge mit der Gruppe. Die Beispiele, die ich im folgenenden verwenden werde, sind nur dazu da, die Funktion des Syntaxes zu erklären, sie sind nicht immer sinnvoll!
Beginnen wir mal:

BEGIN char
IF ~NumTimesTalkedTo(0)~ THEN BEGIN 1

Hier wird die Bedingung für den ersten Dialog festgelegt. *NumTimesTalkedTo* ist eine Variable, die beschreibt, wie oft der Charakter schon einen Dialog geführt hat (Number of times talked to). Die Zahl in der Klammer legt den Wert der Variablen fest. Weiterhin ist *NumTimesTalkedTo* eine Variable, die im Spiel verankert ist und nicht von Ihnen definiert werden muss. Meistens unterscheidet man nur zwischen *NumTimesTalkedTo(0)*, dem ersten Gespräch zwischen der Gruppe und dem Charakter, und weiteren Dialogen. Ein vorgestelltes *!* wirkt auf eine Variable wie ein "NOT", d.h. eine Verneinung. Also ist *!NumTimesTalkedTo(0)* die Bedinung, dass *NumTimesTalkedTo* NICHT den Wert "0" hat. Man kann auch anders zum Ziel kommen: Mit *NumTimesTalkedToGT(0)* setzt

man die Bedinung, dass *NumTimesTalkedTo* einen Wert GRÖSSER ALS (greater than, GT) *0* hat. Bitte nicht mit GRÖSSER GLEICH verwechseln.
Dann machen wir mal mit der nächsten Zeile weiter.

SAY ~Hallo.~

Na gut, da war jetzt nichts schwierig oder neu.

IF ~ReputationGT(Player1,5)~ THEN REPLY ~Hallo, ich bin <GABBER>!~ GOTO 2

Reputation ist wie *NumTimesTalkedTo* eine Variable; sie entspricht dem Ruf. *GT* ist wieder der Zusatz, der nachschaut, ob die Variable grösser ist. Diese Variable hat jetzt zwei Argumente. Das erste bezieht sich auf den Spieler, nach dessen Ruf geschaut wird, das zweite auf den Wert der Variablen. Diese Antwort steht also zur Verfügung wenn der Ruf von *Player1* grösser als *5* ist. *Player1* ist der HC, die anderen "Player" (2-6) werden nach ihrem "Eintrittsdatum" in die Gruppe durchnummeriert (aufsteigend). Da der Ruf für alle Gruppenmitglieder dergleiche ist, könnte man auch den Ruf von *Player3* abfragen. Allerdings weiss man ja nicht, ob die Gruppe schon aus 3 Mitgliedern besteht, also sollte man so etwas lieber lassen. Das zweite neue hier ist *<GABBER>*. In diesen <>-Zeichen stehen System-Variablen in Zeichenketten. *<GABBER>* ist hier die Variable für den Sprecher. Das Programm setzt dafür den Namen desjenigen ein, der den Charakter angesprochen hat. Eine andere dieser Variablen ist *<CHARNAME>*, sie steht für den Namen des HC. Es gibt noch einige weitere dieser Variablen, die aber vor allem dazu dienen, im Englischen zwischen einem männlichen und einem weiblichen HC zu unterscheiden. Im Deutschen sind sie nicht notwendig, da es hier für den männlichen und den weiblichen HC unterschiedliche dialog-Dateien gibt (dialog.tlk/dialogF.tlk). Deren Verwendung wird in der nächsten Antwort gezeigt.

IF ~ReputationGT(Player1,5) Class(Player1,PALADIN)~ THEN REPLY ~Diese Gruppe führt ein Paladin, Herr <CHARNAME>.~ ~Diese Gruppe führt ein Paladin, Frau <CHARNAME>.~ GOTO 3

Zuerst zu den Bedingungen: für diese Antwort müssen zwei erfüllt sein; der Ruf muss über 5 liegen und der HC muss ein Paladin sein. Der Aufbau der Variablen *Class* ist wie der von *Reputation*. Der erste Parameter bezieht sich auf das Gruppenmitglied (*Player1*), der zweite auf den Wert (*PALADIN*). Der Wert einer Variablen muss nicht immer eine Zahl sein.
Schreibt man 2 Antwortzeichenketten hintereinander, dann ist die erste für einen männlichen HC (in die dialog.tlk), die zweite für einen weiblichen HC (in die dialogF.tlk).
Wir benötigen für unseren ersten Dialog noch eine Antwortmöglichkeit, falls die Gruppe einen Ruf von 5 oder niedriger hat. Eine Möglichkeit wäre:

IF ~ReputationLT(Player1,6)~ THEN REPLY ~Haut ab!~ EXIT

Diese Antwort wäre für die Gruppe, die die obigen Antworten nicht wählen kann. Ruf KLEINER ALS (lower than, LT) "6" ist "5" und darunter. Einfacher wäre auch

IF ~~ THEN REPLY ~Haut ab!~ EXIT

Diese Antwort würde immer zur Verfügung stehen.
Es fehlt jetzt noch ein END für unser BEGIN 1 Dieser Dialog sieht jetzt so auf:

BEGIN char
IF ~NumTimesTalkedTo(0)~ THEN BEGIN 1
SAY ~Hallo~
IF ~ReputationGT(Player1,5)~ THEN REPLY ~Hallo, ich bin ¡GABBER¿!~ GOTO 2
IF ~ReputationGT(Player1,5) Class(Player1,PALADIN)~ THEN REPLY ~Diese Gruppe führt ein Paladin, Herr <CHARNAME>.~ ~Diese Gruppe führt ein Paladin, Frau <CHARNAME>.~ GOTO 3

IF ∼∼ THEN REPLY ∼Haut ab!∼ EXIT
END

In diesem Dialog haben wir auf die Dialoge *2* und *3* verwiesen. Diese müssen jetzt geschrieben werden.

IF ∼∼ THEN BEGIN 2

hiermit beginnen wir Dialog *2*

SAY ∼Oh wie schön, darf ich mit Euch reisen <GABBER>?∼
IF ∼∼ THEN REPLY ∼Nein.∼ EXIT
IF ∼∼ THEN REPLY ∼Ja, gerne.∼ DO ∼SetGlobal("CHARJoinedParty","GLOBAL",1) JoinParty()∼
EXIT
END

Die erste Antwort beendet das Gespräch. Bei der zweiten Antwort wird der Charakter in die Gruppe aufgenommen. *JoinParty* ist die Aktion die einen Charakter in die Gruppe aufnimmt. *SetGlobal* ist die Aktion, die den Wert einer Variablen setzt. Die Variable ist *CHARJoinedParty*, ihr Wert wird auf *1* gesetzt. *GLOBAL* bedeutet im Gegensatz zu *LOCALS*, dass die Variable auf für andere Dateien benützt wird (Skripte, andere Dialoge); *LOCALS* bezeichnet Variablen, die nur in dieser Datei benutzt werden. Generell kann man alle Variablen als *GLOBAL* definieren, wenn man auf Nummer sicher gehen will. Wenn für eine Variable kein Wert gesetzt ist, wird angenommen, dass ihr Wert *0* ist. Beim Einführen von neuen Variablen wie hier von *CHARJoinedParty* ist darauf zu achten, dass dieser Variablenname nicht schon vergeben ist. Um dies zu vermeiden sollte ein sehr spezifischer Name gewählt werden, der am besten den Namen des Charakters enthält.
Die Variable wird gesetzt, da man in Zukunft bei Dialogen und Skripten unterscheiden will, ob der Charakter in der Gruppe ist oder eben nicht.
Dialog *3* ist ähnlich:

IF ∼∼ THEN BEGIN 3
SAY ∼Ich wollte immer schon einmal mit einem Paladin reisen, darf ich?∼
IF ∼∼ THEN REPLY ∼Nein.∼ EXIT
IF ∼∼ THEN REPLY ∼Ja, gerne.∼ DO ∼SetGlobal("CHARJoinedParty","GLOBAL",1) JoinParty()∼
EXIT
END

Damit man den Charakter auch später noch aufnehmen kann, benötigen wir noch einige weitere Dialoge.

IF ∼NumTimesTalkedToGT(0) Global("CHARJoinedParty","GLOBAL",0)∼ THEN BEGIN 4

NumTimesTalkedToGT(0) ist die Bedinung, dass es sich nicht um den ersten Dialog handelt. *Global(...)* stellt sicher, dass wir den Charakter nicht schon aufgenommen haben (dann wäre die Variable auf *1* gesetzt worden. Es ist zudem ein Beispiel für die Abfrage der selbst erstellten Variablen. Dass wir bei unserem ersten Dialog nicht abgefragt haben, ob der Charakter sich schon in der Gruppe befindet, liegt daran, dass er sich vor dem ersten Gespräch noch nicht in der Gruppe befinden kann. Dass es das erste ist, haben wir sicher gestellt mit *NumTimesTalkedTo(0)*.

SAY ∼Darf ich jetzt mit?∼
IF ∼∼ THEN REPLY ∼Nein.∼ EXIT
IF ∼∼ THEN REPLY ∼Ja, jetzt schon.∼ DO ∼SetGlobal("CHARJoinedParty","GLOBAL",1) JoinParty()∼ EXIT
END

Dies beschreibt nur die nötigsten Schritte. Ein NPC sollte allerdings geistreichere Dialoge haben!

3.2.4 Die charP.dlg

Diese Datei regelt das Rauswerfen von Mitgliedern einer Gruppe. Wenn man im Spielbildschirm die Gruppe ändert und diesen Charakter entfernt, wird dieser Dialog gestartet.
Die Datei beginnt also mit

BEGIN charP
IF ~Global("CHARKickedOut","GLOBAL",0) Global("CHARJoinedParty","GLOBAL",1)~ THEN BEGIN 1

Die Variable soll sicherstellen, dass der Charakter nicht aus der Gruppe entfernt wurde. Sie ist unbedingt notwendig! Die Funktion kann nicht von der Variablen *CHARJoinedParty* übernommen werden. Zumindest war dies bei mir einmal eine Fehlerquelle. Ich sehe keinen Grund für diese neue Variable, Wes Weimer auch nicht, aber es ging nicht ohne (empirisches Wissen vs. Logik). Mit *CHARJoinedParty* stellen wir sicher, dass ich der Charakter auch in der Gruppe befindet.

SAY ~Ich soll Euch verlassen?~ IF ~~ THEN REPLY ~Nein, ich kann nur meine Maus nicht bedienen.~ DO ~JoinParty()~ EXIT

Nach dem "Entfernen" über den Spielbildschirm gilt ein Charakter als entfernt. Will man das Verhindern muss man ihn praktisch wieder aufnehmen. Dazu die Aktion *JoinParty()*.

IF ~~ THEN REPLY ~Ja, weg mit Euch.~ DO ~SetGlobal("CHARKickedOut","GLOBAL",1) SetGlobal("CHARJoinedParty","GLOBAL",0)~ EXIT END

Nachdem der Charakter aus der Gruppe etngültig entfernt wurde, müssen wir die beiden Variablen auf den richtigen Wert setzen.
Jetzt müssen wir dem Charakter die Möglichkeit geben wieder aufgenommen zu werden:

IF ~Global("CHARKickedOut","GLOBAL",1) Global("CHARJoinedParty","GLOBAL",0)~ THEN BEGIN 2

Wir stellen mit Hilfe der beiden Variablen fest, dass der Charakter aus der Party geworfen wurde.

SAY ~Was denn, soll ich mitreisen oder nicht? Wisst Ihr überhaupt, was Ihr wollt?~
IF ~~ THEN REPLY ~Reist mit uns!~ DO ~SetGlobal("CHARKickedOut","GLOBAL",0) SetGlobal("CHARJoinedParty","GLOBAL",1) JoinParty()~ EXIT
IF ~~ THEN REPLY ~Bleibt wo Ihr seid!~ EXIT
END

Das ganze jetzt noch mal zusammen:

BEGIN charP
IF ~Global("CHARKickedOut","GLOBAL",0)~ THEN BEGIN 1
SAY ~Ich soll Euch verlassen?~
IF ~~ THEN REPLY ~Nein, ich kann nur meine Maus nicht bedienen.~ DO ~JoinParty()~ EXIT
IF ~~ THEN REPLY ~Ja, weg mit Euch.~ DO ~SetGlobal("CHARKickedOut","GLOBAL",1) SetGlobal("CHARJoinedParty","GLOBAL",0)~ EXIT
END

IF ~Global("CHARKickedOut","GLOBAL",1) Global("CHARJoinedParty","GLOBAL",0)~ THEN BEGIN 2
SAY ~Was denn, soll ich mitreisen oder nicht? Wisst Ihr überhaupt, was Ihr wollt?~
IF ~~ THEN REPLY ~Reist mit uns!~ DO ~SetGlobal("CHARKickedOut","GLOBAL",0) SetGlobal("CHARJoinedParty","GLOBAL",1) JoinParty()~ EXIT

IF ∼∼ THEN REPLY ∼Bleibt wo Ihr seid!∼ EXIT
END

Theoretisch kann ein Char auch irgendwohin geschickt werden. Dazu kann man in Dialog *1* noch folgende Antwort hinzunehmen.

IF ∼∼ THEN REPLY ∼Geht in den Tempelbezirk.∼ DO ∼SetGlobal("CHARKickedOut","GLOBAL",1) SetGlobal("CHARJoinedParty","GLOBAL",0) MoveGlobal("AR0900","CHAR",[2029.3136])∼ EXIT

Hierzu haben wir die Aktion *MoveGlobal* verwendet. *AR0900* gibt das Gebiet an (den Tempelbezirk), in das *CHAR* an den Ort [x,y] gehen soll. *x* und it y sind die Koordinaten auf der Karte. Man sollte vor den Benutzung der Koordinaten sicherstellen, dass dort auch ein freier Platz ist (auf dem hier benutzten steht schon MEIN NPC!). *CHAR* ist die DeathVariable (DV) des Charakters, diese ist der spielinterne "Name" des Charakters. Er muss nicht identisch mit dem normalen Namen des Charakters sein, da der normale Name manchmal ja länger als 8 Zeichen sein soll. Dadurch können zwei unterschiedlich Charaktere denselben Namen haben, ihre DV muss aber unterschiedlich sein. Wie weiter unten (siehe **??**) beschrieben gibt es auch andere Aktionen für den Gebietswechsel. Ach ja, noch für die Korrekten. Die im Spiel enthaltenen NPCs können aus der Unterwelt nicht in ein anderes Gebiet gehen. Wenn man also das nachemfpinden will, muss für die obere Antwort die Bedingung rein, dass es sich nicht um eine Unterweltkarte handeln darf. Eine einzige Bedinung dafür gibt es nicht. Man muss JEDE Unterweltkarte überprüfen. Die Daten holt man sich aus den *P.dlg-Dateien der Original-NPCs.

3.2.5 Die charJ.dlg

Noch was Allgemeines vorne weg: Die Zuordnung von Dialogen in bestimmte Dateien ist willkürlich. Der Solaufein-Romanzen-Mod hat nur eine einzige Dialog-Datei. Das von mir verwendete System ist so nahe wie möglich an den Dialogdateien der Original-NPCs. Diese Datei enthält Dialoge, mit denen der Charakter im Spiel interagiert, wenn er in der Gruppe ist (J = Joined). Dazu gibt es mehrere Möglichkeiten:

- einen Dialog mit dem HC führen
- sich in einen anderen Dialog einmischen
- in einem Dialog zusätzliche Antwortmöglichkeiten oder Trigger einfügen

Dialoge mit dem HC Die Dialoge in der charJ.dlg werden über das Skript (siehe **??**) des Charakters mit dem Befehl *StartDialogue()* gestartet. In der Klammer gibt man die DV des Charakters in Anführungszeichen an. Für den HC hiesse der Syntax:

Startdialogue(Player1)

für Nalia z.B.:

Startdialogue("Nalia")

Der Dialog wird ganz normal wie oben beschrieben (siehe **??**) geführt.

Einmischen in einen anderen Dialog WeiDU ermöglicht es einem, sich elegant und mit wenigen Befehlen in einen Dialog von anderen einzumischen. Technisch läuft es so ab, dass nach einer Dialogzeile der Verweis auf die nächste Dialogzeile geändert wird, so dass er auf die einzufügende Dialogzeile verweist. Diese endet dann mit einem Verweis auf die eigentliche Fortführung des Dialoges.
In WeiDU sieht das dann wie folgt aus (Beispiel aus dem Readme von WeiDU, selber noch nicht ausprobiert):

INTERJECT_COPY_TRANS IDRYAD1 1 MinscDryad
== MINSCJ IF ~IsValidForPartyDialog(Minsc")~ THEN
~Boo ist entsetzt, dass dieser komische Magier diese netten Damen gefangenhält! Können Minsc und Boo helfen?~
== IDRYAD1 IF ~IsValidForPartyDialog(Minsc")~ THEN
~He Sterblicher, das ist gerade eine dramatische Szene, bitte unterbrich uns nicht!~
END IDRYAD2 1

Dies ist eine Unterbrechung des Dryadendialogs im Irenicus Kerker durch Minsc. *INTERJECT_COPY_TRANS* ist der Befehl, der alles auslöst. *IDRYAD1* ist die Dialogdatei dessen, der spricht. *1* ist die Nummer des Dialogs in der *IDRYAD1.DLG*. *MinscDryad* ist der Name (anstatt einer blossen Nummer) des folgenden Dialoges. *== MINSCJ* gibt an, in wessen *.dlg-Datei der folgende Text geschrieben werden soll. *IF ~IsValidForPartyDialog(Minsc")~* stellt sicher, dass Minsc auch in der Gruppe ist, da nur der Dialog der Dryaden geändert wird. Das gleiche macht dann *== IDRYAD1 IF ~IsValidForPartyDialog(Minsc")~*. Nach dem *END* kommt dann der Name der Dialogdatei (*IDRYAD2*) mit Dialognummer (*1*), bei dem es weitergehen soll. Die Antwort der Dryade kann man auch weglassen, dann würde der Dialog einfach weitergehen, ohne dass die Dryaden auf Minsc's Einwurf eingehen würden.

Zusätzliche Antwort(en) Dies wird z.B. benötigt, um die Antworten des Geistes im Einsprengsel so zu ergänzen, dass auch der neue NPC herbeigerufen werden kann. Genau dieses Beispiel werden ich hier auch verwenden. Man kann diese zusätzlichen Antworten, mit denen andere *.dlg-Dateien modifiziert werden auch in eine beliebige *.d-Datei schreiben.

EXTEND_TOP FATESP 6

Mit *EXTEND_TOP* wird angegeben dass die neue Antwortmöglichkeit als erste aufgeführt werden soll. Wenn man diese Stelle nicht mag bleibt einem noch *EXTEND_BOTTOM*; dieser Befehl fügt die Antwort als letzte Antwort ein. Andere Positionen sind nicht möglich. *FATESP* ist der Name der *.dlg-Datei in die die Antwort eingefügt werden soll (also *FATESP.DLG*). Die *6* bezeichnet die Nummer des Dialoges, der mit der folgenden Antwort ergänzt werden soll. Um diese Nummer zu finden, sollte die Dialog-Datei mittels WeiDU in eine *.d-Datei umgewandelt werden (wie siehe **??**). Dann folgt die Zeile, die eingefügt werden soll:

IF ~!InParty("DV des Charakters") !Dead("DV des Charakters") Gobal("CharSummoned","GLOBAL",0)~
THEN REPLY Text DO ~SetGlobal("CharSummoned","GLOBAL",1)~ EXTERN FATESP 8
END

Zuerst erläutere ich mal die Bedinungen für die Dialogzeile:

- *!InParty("DV des Charakters")* stellt fest, dass der Charakter sich nicht schon in der Gruppe befindet (wenn ja, sollte man ihn nicht herbeirufen können). *InParty()* frägt ab, ob sich die Person in der Gruppe befindet; das vorgestellte *!* ist die Verneinung dieser Aussage (für die Programmierer: ändert den Boolean-Wert der Variablen).
- *!Dead("DV des Charakters")* stellt sicher, dass der Charakter nicht schon gestorben ist. Eine Wiederbelebung kann an dieser Stelle natürlich auch erlaubt werden (dann die Bedingung

einfach weglassen), ist aber bei allen anderen NPCs nicht der Fall. Wiederum ist durch das *!* die Frage, ob der Charakter tot ist (*Dead()*) verneint worden.

- *Gobal("CharSummoned","GLOBAL",0)* stellt einfach fest, dass der Charakter während des Dialoges noch nicht herbeigerufen wurde. Der Wert dieser Variablen wird nach der Antwortzeile dann auch auf *1* gesetzt und ist dann für das Skript, das die Herbeirufung durchführt (siehe **??**), wichtig.

Danach kommt der *Text* der Antwort, also in etwa: "Bringe Charaktername zu mir." Als Reaktion auf die Antwort wird der Wert der Variablen erhöht. Dann kommt der Verweis auf Dialog-Datei und Dialognummer (wieder in der *.d-Datei nachschauen, wo die anderen Antworten hinverweisen), mit der es dann weitergehen soll *EXTERN FATESP 8*. Das ganze wird mit *END* abgeschlossen. Wie alle Sachen, die mit *APPEND* anfangen, muss dieser Einschub am Ende der Datei stehen.
Wenn allerdings der Verweis auf den folgenden Dialog kompliziert ist und man nicht weiss, ob man dadurch den Ablauf durcheinander bringt (wie zum Beispiel bei dem Dialog in der Hölle, bei dem alle NPCs dem HC ihre Loyalität versichern), dann sollte man eine andere WeiDU-Konstruktion benutzen:

IF ∼∼ THEN BEGIN Dialognummer
SAY ∼Text∼
COPY_TRANS FATESP 6
END

3.2.6 Die Bchar.dlg

Diese Datei ist für die Interaktion des NPC mit anderen Gruppenmitliedern (auch dem HC) und ist nicht unbedingt erforderlich. Nicht-NPCs fangen mit dieser Datei nichts an. Diese Dialoge werden über das Skript (siehe **??**) mit dem Befehl *Interact("DV des Gruppenmitgliedes")* gestartet.
Zuerst werde ich ein Gespräch zwischen unserem Charakter und einem anderen Mitglied der Gruppe (also zwei Personen) behandeln. Als Beispiel werde ich einen Dialog mit Nalia führen. Der Anfang ist schon bekannt:

BEGIN Bchar

Dann schauen wir, ob der Charakter, mit dem unser NPC sprechen soll, überhaupt in der Gruppe ist und dieser Dialog nicht schon stattgefunden hat.

IF ∼IsValidForPartyDialog("Nalia") Global("CHARtalksNalia","LOCALS",0)∼ THEN BEGIN 1begin

IsValidForPartyDialog fragt nach, ob der Charakter in der Klammer Mitglied der Gruppe ist. Der abgefragte Name ist die DV. *1begin* ist der Name des Dialogs. Die Variable, die nachher auf *1* gesetzt wird, soll sicherstellen, dass der Dialog nur einmal gesprochen wird. Wenn man den Dialog gezielt starten will kann man hier noch alternativ eine weitere Variable abfragen, die dann durch das Skript gesetzt wird (siehe **??**).

SAY ∼Hübsche Frisur, Nalia.∼
IF ∼∼ THEN DO ∼SetGlobal("CHARtalksNalia","LOCALS",1)∼ EXTERN BNALIA chartalk1
END

Ohne eine Bedingung ∼∼ geht es weiter. Wir setzen die Variable, dass der Dialog stattgefunden hat. Danach sage wir mit *EXTERN*, dass es in einer anderen Datei mit dem Dialog chartalk1 weitergeht. Diese Datei soll *BNALIA* sein, Nalia's Bchar.dlg. Wie die Bchar.dlg eines Charakters heisst, kann man mit NearInfinity (NI) feststellen. Dazu öffnet man die interne Datei *interdia.2DA*. Dort steht hinter der DV der Name der Bchar-Datei (und wenn man TdB installiert hat, noch der

Name der Bchar25-Datei). Als nächstes müsste man nun in die *BNALIA.DLG* Nalia's Antwort einbringen. Dies geschieht mit folgendem Syntax:

APPEND BNALIA
IF ~~ THEN BEGIN chartalk1

APPEND Datei fügt der Datei *Datei.dlg* den nachfolgenden Syntax hinzu. Dabei muss darauf geachtet werden, dass *APPEND* wie eine neue *.dlg Datei (*BEGIN Datei*) wirkt und deshalb an das Ende der *.d-Datei gesetzt werden sollte. Ohne eine Bedinung ~~ lassen wir jetzt Nalia ihren Spruch aufsagen.

SAY ~Danke!~
IF ~~ THEN EXIT
END
END

Das erste *END* gehört zu dem *BEGIN*, das zweite beendet den Einschub *APPEND*.
Nun wollen wir wahrscheinlich auch mal längere Dialoge machen. Das wäre mit dem obigen Modell sehr lästig, deshalb gibt es eine elegantere Methode. Wir verwenden denselben Einstieg in den Dialog (bis *...chartalk1 END*). Danach benutzen wir eine Dialogkette (*CHAIN*).

CHAIN BNALIA chartalk1

Dieser Syntax sagt, dass es in der Kette mit der Datei *BNALIA* weitergeht und der Dialog *chartalk1* ist. Dann folgt Nalia's Text:

~Findest du. Ich war schon lange nicht mehr beim Friseur!~
== Bchar
~Doch, du kämmst sie sicherlich jeden Tag~

== zeigt an, dass es mit dem Text in der nächsten Datei (*Bchar*) weitergeht, dann kommt der Text von char. Und so geht es in einem fort.

== BNALIA
~Da hast du allerdings recht. Ich finde man sollte ich zumindest morgens mal die Haare kämmen.~
== Bchar
~Leider ist das, wenn man auf Reisen ist, manchmal recht schwierig.~

So langsam wollen wir aber zum Schluss kommen. Also beenden wir mit dem nächsten Beitrag die Kette.

== BNALIA
~Ja, da stören manchmal so aufdringliche Monster, die einfach keinen Sinn für die Notwendigkeit von Körperpflege haben.~
END Bchar 1end

END beendet die Kette. Danach wird darauf verwiesen, wo es weitergeht (nämlich in der Datei *Bchar* mit dem Dialog *1end*. Eine Kette kann nicht direkt beendet werden, sie muss in einen Dialog zurückgeführt werden. Genauso braucht es einen Anfangsdialog, der in die Kette führt. Einen Zweizeiler kann man also nicht mit einer Kette ausdrücken. Das Ende des Dialogs haben wir dann wieder in der Datei *Bchar*:

IF ~~ THEN BEGIN 1end
SAY ~Du sagst es, Männer sind allerdings manchmal auch nicht besser als Monster.~
IF ~~ THEN EXIT
END

Das Ende würde man auch in den *APPEND*-Block schreiben, wenn Nalia das "letzte Wort" haben soll. Zum Schluss noch mal alles in einem Guss:

BEGIN Bchar
IF ~IsValidForPartyDialog("Nalia") Global("CHARtalksNalia","LOCALS",0)~ THEN BEGIN 1begin
SAY ~Hüsche Frisur, Nalia.~
IF ~~ THEN DO ~SetGlobal("CHARtalksNalia","LOCALS",1)~ EXTERN BNALIA chartalk1
END
CHAIN BNALIA chartalk1
~Findest du. Ich war schon lange nicht mehr beim Friseur!~
== Bchar
~Doch, du kämmst sie sicherlich jeden Tag~
== BNALIA
~Da hast du allerdings recht. Ich finde man sollte ich zumindest morgens mal die Haare kämmen.~
== Bchar
~Leider ist das, wenn man auf Reisen ist, manchmal recht schwierig.~
== BNALIA
~Ja, da stören manchmal so aufdringliche Monster, die einfach keinen Sinn für die Notwendigkeit von Körperpflege haben.~
END Bchar 1end
IF ~~ THEN BEGIN 1end
SAY ~Du sagst es, Männer sind allerdings manchmal auch nicht besser als Monster.~
IF ~~ THEN EXIT
END

Ein Dialog mit mehreren Teilnehmern läuft ähnlich ab. Genaueres folgt in späteren Versionen.

3.2.7 Die char25P.dlg

Hier ist es schon ein bisschen schwerer, da man darauf achten muss, in welchem Gebiet man sich befindet.
Fangen wir mal an:

BEGIN char25P
IF ~Global("CHARKickedOut","GLOBAL",0) THEN BEGIN 1

den kannten wir schon.

SAY ~Nein, ich will nicht weg!~
IF ~~ THEN REPLY ~Dann bleib halt.~ DO ~JoinParty()~ EXIT

Jetzt wird es schwieriger:

IF ~AreaCheck("AR4500")~ THEN REPLY ~Bleibt hier im Einsprengsel.~ DO ~SetGlobal("CHARKickedOut","G
SetGlobal("CHARJoinedParty","GLOBAL",0) MoveToPointNoInterrupt([2212.1349]) Face(14)~
EXIT

AreaCheck ist die Bedingung, dass die Gruppe sich in dem Gebiet in der Klammer befindet (hier die *AR4500*). *AR4500* ist das Einsprengsel. Die Aktion *MoveToPointNoInterrupt([x.y])* legt fest, dass der Charakter zu dem Punkt x,y läuft. *Face(..)* legt die Richtung fest, in die er dann schaut. Wie die Zahlen in der Klammer mit den Richtungen verknüpft sind, kann ich nicht sagen; das müssen Sie halt ausprobieren.

IF ~!AreaCheck("AR4500") !AreaCheck("AR4000") !AreaCheck("AR6200")~ THEN REPLY ~Geht in das Einsprengsel und wartet dort.~ DO ~SetGlobal("CHARKickedOut","GLOBAL",1) SetGlo-

bal("CHARJoinedParty","GLOBAL",0) CreateVisualEffectObject("SPDIMNDR",Myself) Wait(2) MoveBetweenAreas("AR4500",[2212.1349],14)~ EXIT

Hier wird überprüft, dass sich der Charakter nicht im Einsprengsel befindet und nicht in zwei anderen Gebieten, von denen er nicht wegkommen soll (*AR6200* ist z.B. der Endkampf). Die Aktion *CreateVisualEffectObject* lässt eine grafische Animation ablaufen, die durch erste Argument definiert und durch das zweite platziert (am Ort, wo sich *Myself* gerade befindet) wird. Mehr kann ich nicht dazu sagen; man kann diesen Effekt auch weglassen. *Wait* lässt zwei Zeiteinheiten (Sekunden ?) zwischen den Aktionen verstreichen. *MoveBetweenAreas* ist wieder ein Befehl, der den Charakter in ein anderes Gebiet bringt. Die Argumente dieser Aktion sollten inzwischen bekannt sein.

IF ~!AreaCheck("AR4500") !AreaCheck("AR4000") !AreaCheck("AR6200")~ THEN REPLY ~Wartet hier auf mich, ich bin gleich wieder da.~ DO ~SetGlobal("CHARKickedOut","LOCALS",1) SetGlobal("CHARJoined
EXIT
END

3.2.8 Die Bchar25.dlg

Diese Datei "missbrauchen" wir mal für den Dialog im Einsprengsel.

BEGIN Bvanim25
EXTEND_ TOP FATESP 6

Dies fügt dem Dialog *6* der Datei *FATESP.DLG* folgende Zeilen hinzu:

IF ~!InParty("CHAR") !Dead("CHAR") Global("CharSummoned","GLOBAL",0)~

Die Bedingung für die Dialogoption ist, dass der Charakter nicht in der Gruppe ist (*!InParty*) und auch nicht schon gestorben ist (*!Dead*). *CHAR* ist hier die DV. Auch soll der Charakter noch nicht herbeigerufen worden sein.

THEN REPLY ~Bringt mir Char, meinen treuen Begleiter~ DO ~SetGlobal("CharSummoned","GLOBAL",1)~
EXTERN FATESP 8

Jetzt wird die neue Antwortoption eingeführt. Wenn die angewählt wird, wird der Wert der Variablen geändert. Danach wird zum nächsten Dialog (*8*) weitergeleitet.

END

Das obligatorische *END*.

3.2.9 Die char25J.dlg

Die char25J.dlg wird analog zu der charJ.dlg (siehe **??**) erstellt.

3.2.10 Für mehrsprachige Mods - *.tra-Dateien

Möchte man seinen NPC in mehreren Sprachen anbieten, kann man zum einen alle *.d-Dateien übersetzen und dann verschiedene Downloads zur Verfügung stellen. Man kann allerdings auch viele Sprachen in einer *.d zur Verfügung stellen. Dazu dienen die *.tra-Dateien. Man benützt sie, indem in einer *.d einen Verweis auf einen Zeichnsatz anbringt und den Zeichensatz in eine *.tra-Datei schreibt. Für jede Sprache gibt es dann eine *.tra-Datei. Beispiel:
in der char.d Datei:

IF ~~ THEN BEGIN 1 SAY @1

@1 ist der Verweis

benötigt die char.tra Datei:

@1 = ∼Hallo!∼

die englische char.tra Datei:

@1 = ∼Hello!∼

Welche char.tra Datei (englisch/deutsch) benutzt werden soll, wird in der *.tp2-Datei (siehe **??**) angegeben.

4 Die Skripte

4.1 Welche Skripte werden benötigt

Man braucht zuerst ein Skript, das den Charakter erschafft. Danach benötigt man ein Skript für den Charakter. Wenn es sich bei dem Charakter nicht um einen NPC handelt, benötigt er noch ein Skript für seine Kampfhandlungen; der NPC benötigt ein solches nicht, da er ja von dem Spieler gesteuert wird.

4.2 Wie erstelle ich ein Skript

Skripte können in einem beliebigen Editor geschrieben werden und werden dann mit der Endung .bcs abgespeichert. Danach startet man NearInfinty (NI). Unter dem Menüpunkt *Game* findet man *Open file* und wählt es an. Im daraufhin erscheinenden Fenster wählt man *Open external file*, da es nicht um eine Datei aus dem Spiel handelt. Mit Hilfe von *Browse* findet man dann die Datei und öffnet sie in einem neuen Fenster. Danach findet man im oberen Teil des Fensters den Skript-Text. Dieser wird mittels copy'n paste (Strg+c, Strg+v) in die untere Hälfte kopiert. Danach wählt man unter links *Compile*. Tritt jetzt eine Fehlermeldung auf, ist der Syntax nicht korrekt. Wenn alles geklappt hat wird mittels *Save* die Datei gespeichert und fertig ist das Skript. Sollte sich das *Save*-Feld nicht rechts unten befinden muss das Fenster vergrössert werden, dann erscheint das Feld dort schon. Will man eine der spielinternen Dateien anschauen, kann man die Datei im *Open file*-Fenster unter *Open Internal File* auswählen.

4.3 Wie sieht ein Skript aus

Ein Skript besteht aus einer Bedinung *IF xxx* und den Folgen *THEN RESPONSE #100 xxx END*. *xxx* steht für beliebig viele Folgen/Bedingungen. Die Zahl hinter # gibt an mit wieviel prozentiger Wahrscheinlichkeit die Folgen eintreten. Man kann also mehrere *RESPONSE #xx* angeben. Allerdings sollten sich die Prozente wieder zu 100 aufaddieren. Noch zur Form: Wenn ich hier eine neue Zeile begonnen habe, muss auch im Skript-Text eine neue Zeile begonnen werden.

4.4 Das Erschaffungsskript

Dieses Skript muss nachher in die *ARxxxx.bcs* (*xxxx* steht für die Nummer des Gebietes) des Gebiets eingefügt werden, in dem der Charakter stehen soll (wie wird unter **??** erklärt). Das Skript nennen wir mal *Archar.bcs*.
Das Skript selbst sieht wie folgt aus:

IF
Global("CHARExists","AR0900",0)

Diese Bedingung prüft, ob der Charakter schon in diesem Gebiet erschaffen wurde. Der Variablenname ist *CHARExists*, der Wert *0* und *AR0900* ist wieder der Tempelbezirk und bezeichnet den Gültigkeitsbereich der Variablen.

THEN
RESPONSE #100

SetGlobal("CHARExists","AR0900",1)
Nun haben wir eine 100% -tige Antwort und setzen die Variable auf *1*, womit wir festlegen, dass der Charakter jetzt existiert. Die Erschaffung folgt in der nächsten Zeile.

CreateCreature("CHAR",[2029.3136],3)
END

Mit *END* wird der Syntax abgeschlossen. Mit *CreateCreature* wird die Kreatur erschaffen. *CHAR* ist der Name der *.cre Datei, die erschaffen werden soll; *[x.y]* ist wieder der Ort und *,z* die Richtung (siehe **??**).

4.5 Das Charakterskript

Hier werden alle benötigten Auslöser (Trigger) und Handlungen des Charakters hineingeschrieben.

4.5.1 EP-Anpassung

Wir benutzen dieses Skript, um dem Charakter genügend Erfahrungspunkte zukommen zu lassen, damit er in der Gruppe nicht hinterherhängt.

IF
Global("CHARGetXP","GLOBAL",0)
XPGT(Player1,300000)

Hier haben wir wieder zuerst eine Variable abgefragt, um zu verhindern, dass der Charakter permanent Erfahrungspunkte bekommt. *XP* ist wieder ein Variable, die die Erfahrungspunkte eines Charakters beschreibt. *Player1* bezieht sich wieder auf den Charakter, nach dessen Erfahrungspunkten gefragt wurde. Die Zahl danach ist der Wert den die Variable haben muss (*300.000*). Das GT legt fest, dass der Charakter mehr als 300.000 Erfahrungspunkte haben soll, damit die Folgen eintreten.

THEN
RESPONSE #100
AddXPObject(Myself,200000)
Continue()
END

Und wieder wollen wir nur eine Folge (*#100*). *AddXPObject* fügt dem in der Klammer benannten Charakter die an zweiter Stelle genannten Erfahrungspunkte hinzu. *Myself* bezeichnet den Charakter, um dessen Skript es sich hier handelt. *Continue()* bedeutet, dass das Skript hier nicht abbrechen soll, sondern mit dem nächsten Syntax fortfahren.

IF
Global("CHARGetXP","GLOBAL",0)
XPGT(Player1,500000)
THEN
RESPONSE # 100
AddXPObject(Myself,200000)
Continue()
END

Das war jetzt nur dasselbe wie oben, für Gruppen, die schon was erlebt haben.

IF

Global("CharGetXP","GLOBAL",0)
XPGT(Player1,1000000)
THEN
RESPONSE #100
AddXPObject(Myself,500000)
Continue()
IF
Global("CHARGetXP","GLOBAL",0)
THEN
RESPONSE #100
SetGlobal("CHARGetXP","GLOBAL",1)
END

Falls der HC wirklich sehr erfahren ist, gibt es hier nochmals Erfahrungspunkte (500.000). Dann ist aber Schluss und wir setzen die Variable, dass die Erhöhung der Erfahrungspunkte abgeschlossen ist.

4.5.2 Skript, um Dialoge zu starten

Man will ja nicht immer, dass sofort ein Dialog startet, wenn man den neuen NPC in seine Gruppe aufnimmt. Dadurch muss man gewisse Auslöser (Trigger) für den Dialog definieren. Das kann das Betreten eines Gebietes, das setzen einer Variable durch eine Handlung oder einen anderen Dialog oder einfach das Verstreichen von Zeit sein.
Zuerst mal ein allgemeines Beispiel eines Dialoges, der sozusagen als Begrüssung nach der Aufnahme in die Gruppe gestartet wird.

IF
InParty(Myself)
IsValidForPartyDialog("DV des Gesprächspartners")
Global("X#WirUnterhaltenUns","GLOBAL",0)
THEN
RESPONSE #100
SetGlobal("X#WirUnterhaltenUns","GLOBAL",1)
Interact("DV des Gesprächspartners")
END

Die erste Bedingung (*InParty(Myself)*) legt fest, dass derjenige, um dessen Skript es sich handelt in der Gruppe ist. Danach wird geschaut, ob der Gesprächspartner überhaupt für ein Gespräch zur Verfügung steht (*IsValidForPartyDialog("DV des Gesprächspartners")*), denn er könnte ja schlafen oder sich gerade mit jemandem anderen unterhalten. Allerdings funktioniert diese Abfrage nicht mit *Player1*. Für ihn muss man *InParty(Player1)* und *!CheckStat(Player1,STATE_SLEEPING)* abfragen. Die nächste Variable dient dazu, dass der Dialog nur einmal gestartet wird, deshalb wird sie auf einen anderen Wert gesetzt. Sie kann auch im Dialog mit diesem Wert abgefragt werden, um einen bestimmten Dialog zu starten. *Interact()* startet dann den Dialog, der sich in der Bchar.dlg befinden muss. Soll ein Dialog aus der charJ.dlg benutzt werden, ist der Dialog mit *StartDialogue()* zu beginnen.
Wenn man nun möchte, dass das Gespräch nach Betreten eines Gebietes gestartet wird, muss man zu den Bedingungen unter *IF* noch eine weitere hinzufügen:

AreaCheck("ARxxxx")

wobei *xxxx* für die Nummer des gewünschten Gebietes steht. Da es immer eine gewisse Ladezeit beim Betreten eines neuen Gebietes gibt, empfiehlt es sich, zu den Aktionen (nach *RESPONSE #100*) noch ein kurzes *Wait(x)* einzufügen, dass *x* Sekunden wartet, bevor der Dialog beginnt.

Wenn man den Dialog nach einer längeren Zeit erst starten will, sollte man das nicht mit *Wait(x)* machen, da während des *Wait*-Befehls das Skript des Charakters angehalten ist. Hierfür ist eine "Zeitschaltuhr" zu benutzen (*SetGlobalTimer*), dessen Benutzung im folgenden erklärt wird:

IF
InParty(Myself)
InParty("DV des Gesprächspartners")
Global("NurEinmalTimerSetzen","LOCALS",0)
THEN
RESPONSE #100
SetGlobal("NurEinmalTimerSetzen","LOCALS",1)
SetGlobalTimer("X#NameDesTimers","GLOBAL",ONE_DAY)
END

Nachdem sichergestellt wurde, dass alle relevanten Leute in der Gruppe sind, und die Uhr noch nicht gesetzt wurde, wird diesselbe dann gesetzt mit dem Befehl *SetGlobalTimer*. Danach steht der Variablenname der Uhr, dann ob sie *GLOBAL* (also im ganzen Spiel abfragbar) oder *LOCALS* (nur in diesem Skript verwendet) ist. Am Schluss steht dann die Zeit, wann die Uhr abgelaufen ist. *ONE_DAY* hat den Wert eines Tages in Spielzeit. Man kann auch mehrere Tage angeben (z.B. *FOUR_DAYS*). Wenn man ein etwas feineres Raster bevorzugt, dann kann man auch die Zeit in Sekunden (in Realzeit) angeben, wobei 7200 Sekunden einem Tag entsprechen, bzw. 6 Sekunden einer Runde. Zumindest in der Theorie; in der Praxis hängt das wohl von der Rechnerkonfiguration und den Spieleinstellungen ab. Nachdem die Uhr gestellt ist, muss man nachschauen, wann sie abgelaufen ist. Das sieht dann wie folgt aus:

IF
InParty(Myself)
IsValidForPartyDialog("DV des Gesprächpartners")
GlobalTimerExpired("X#NameDesTimers","GLOBAL")
Global("NurEinmalTimerSetzen","LOCALS",1)
THEN
RESPONSE #100
SetGlobal("NurEinmalTimerSetzen","LOCALS",2)
Interact("DV des Gesprächpartners")
END

Zuerst wird überprüft, ob der Char nicht in der Zwischenzeit hinausgeworfen wurde, dann wird geschaut, ob der Gesprächspartner zu Verfügung steht. Dann wird mit dem Syntax *GlobalTimerExpired* abgefragt, ob die Uhr abgelaufen ist. Danach kommt die Variable, die angschliessend wieder hochgesetzt wird, damit der Dialog nur einmal stattfindet. Man kann dafür natürlich auch eine neue Variable einfügen, aber ich bin halt ein Freund des Variablenrecyclings. Weitere Bedinungen können sowohl beim Stellen der Uhr als auch beim Starten des Dialoges eingeführt werden.

4.6 Das Skript für das Einsprengsel

Dieses Skript muss an das AreaSkript des Einsprengsels (*AR4500.BCS*) mit Hilfe der *.tp2-Datei angehängt werden.
IF
Global("CharSummoned","GLOBAL",0)
InParty("CHAR")

Diese Bedinungen sind erfüllt, wenn der Charakter noch nicht herbeigerufen wurde (die Variable ist auf *0*) gesetzt und er sich schon in der Gruppe befindet (*InParty*).

THEN

RESPONSE #100
SetGlobal("CharSummoned","GLOBAL",2)
Continue()
END

Die Reaktion ist, dass die Variable auf den Wert *2* gesetzt wird und es weiter geht (*Continue*).

IF
Global("CharSummoned","GLOBAL",1)
Global("CharSpawnPlane","GLOBAL",0)

Hier ist die Bedingung erfüllt, wenn der Charakter herbeigerufen wurde (die Variable wird in dem Dialog auf *1* gesetzt), er aber noch nicht erschienen ist (die andere Variable ist *0*).

THEN
RESPONSE #100

Jetzt kommen mehrere Antworten

CreateVisualEffect("SPPORTAL",[1999.1218])

Erschafft einen grafischen Effekt (*SPPORTAL*) an der Stelle *[x.y]*

Wait(2)

Wartet zwei Einheiten (Sekunden glaube ich)

CreateCreature("CHAR",[1999.1218],0)

Erschafft den Charakter, wie schon zuvor im Erschaffungsskript am Ort *[x.y]*

AddXPObject("CHAR",2400000)

Dieser Befehl weist *CHAR* (DV) 2.400.000 Erfahrungspunkte zu, damit er ungefähr in der Nähe der anderen Charaktere liegt.

SetGlobal("VanimSpawnPlane","GLOBAL",1)

Der letzte Befehl setzt jetzt die Erscheinungsvariable auf *1*.

END

4.7 Das Kampfskript eines Nicht-NPCs

Dieses ist für einen NPC nicht von Nöten und wird somit erst später geschrieben; wenn überhaupt, da es sehr umfangreich und kompliziert sein wird.

5 Die Fertigstellung der *.cre-Datei

Unsers *.cre-Datei aus Abschnitt 1 fehlen noch ein paar Dinge. Dazu müssen wir das Charakterskript in den "override"-Ordner in Ihrem BG2-Verzeichnis kopieren (nachdem es mit NI bearbeitet wurde). Auch müssen wir die char.dlg-Datei erstellen. Dazu öffnen wir ein DOS-Fenster und gehen in das Verzeichnis, in das wir WeiDU installiert haben, und tippen den Befehl *weidu char.d* ein. Danach sollten wir dort die Datei *char.dlg* finden und ebenfalls in den *override*-Ordner kopieren. Natürlich haben wir vorher die *char.d-Datei* in das WeiDU-Verzeichnis kopiert.
Wenn wir NI noch geöffnet haben, führen wir unter *Game* die Anweisung *Refresh Tree* aus. Danach kennt NI die neuen Dateien im *override*-Ordner. Ansonsten reicht es, NI neu zu starten. Dann öffne wir die *.cre-Datei. Dort wählen wir links oben *edit*. Wir lassen alles so, wie es ist, und ändern nur,

was in diesem Tutorial beschrieben wird. Es empfiehlt sich, das Fenster breiter zu machen, damit man die Beschreibung in der Spalte *Attribute* vollständig lesen kann (Besitzer eines 12"-Displays haben jetzt vielleicht ein Problem).

5.1 Die Änderung des Status

Da wir unseren Charakter als Multiplayer-Charakter erstellt haben, ist er noch als solcher definiert. Dadurch verwendet er nicht die ihm im folgenden (siehe **??**) zugewiesenen Dialoge, sondern den Standard-Multiplayer-Dialog. Dazu ändern wir das Attribut *Flags*, das sich fast am Anfang befindet. Es hat den Wert *(Export allowed(11))*. Wir öffnen jetzt die *.cre Datei eines anderen NPCs (z.B. die *Korgan10.cre*) und kopieren dort mit Hilfe der rechten Maustaste das Attribut. Dazu klicken wir mit der rechten Maustaste auf das Attribut und wählen *copy value* aus. Dann klicken wir wieder mir rechts auf das Attribut in der *.cre-Datei unseres Characters und wählen *replace value*. Jetzt sollte das Attribut den Wert von Korgan haben (*(No flags set)*). Dann gehts zum nächsten Schritt.

5.2 Die Einbindung von alternativen Bildern

Wenn man andere als die vorhandenen Bilder benutzen möchte, sollte man diese in das *override*-Verzeichnis des BG2-Verzeichnisses kopieren. Man benötigt ein kleines Bild und am besten auch ein grosses Bild. Diese kann man entweder selbst erstellen, oder sich aus dem Internet runterladen. Zu den Formaten werde ich mich hier nicht äussern. Es ist im Handbuch beschrieben und die meisten Bilder sind schon im richtigen Format. Dem Charakter ordnet man das kleine Bild unter dem Attribut *Small Portrait*, das grosse unter *Large Portrait* zu. Die Zuordnung erfolgt unter Anwählen des Attributes; dann erscheint unten in der Mitte ein Menü, indem die Datei ausgewählt werden kann. Mittels *Update* wird die neue Datei zugewiesen.

5.3 Die Einbindung der Skripte

Arbeitet man sich weiter nach unten findet man nach den Sound-Effekten die Skript-Attribute. Man weisst dem *General script* das Charakterskript zu. Bei einem Nicht-NPC weisst man dem *Override script* das Kampfskript zu. Wie die Zuweisung funktioniert s.o.

5.4 Die Angabe der Death-Variablen (DV)

Als nächstes muss man unter dem Attribut *Death Variable* den spielinternen Namen des Charakters eintragen. Dazu wählt man die Zeile an, und löscht mit der *Backspace*-Taste den bisherigen Namen (wahrscheinlich *none*), um anschliessend den gewünschten Namen einzutragen (sicherheitshalber sollte er höchtens acht Zeichen lang sein).

5.5 Die Einbindung der Dialog-Datei

Noch weiter unten befindet sich das *Dialog*-Attribut. Diesem wird die char.dlg Datei zugewiesen. Und damit kann die *.cre-Datei abgespeichert werden.

6 Die Installation mit der *.tp2-Datei

Es wird davon ausgegangen, dass sich die Dateien des Mods in dem Ordner *char* befinden.
Man schreibe folgende *.tp2-Datei in einem beliebigen Editor.

BACKUP ~char/char-backup~

BACKUP gibt den Pfad an, unter dem die Kopien der Dateien, die verändert werden, gespeichert werden. Damit kann man später das Mod deinstallieren.

AUTHOR ~meinname@meinprovider~

Hier soll die email-Adresse des Autors stehen, falls ein Benutzer des Mods Fragen hat, oder die Dateien fehlerhaft sind.

LANGUAGE ~Deutsch~ ~deutsch~ ~char/deutsch/setup.tra~

Diese Zeile benötigt man nur, wenn man *.tra-Dateien verwendet. Der erste Parameter gibt den Namen der Sprache an, der zweite den Namen des Verzeichnisses, indem die *.tra-Dateien stehen, und der dritte steht für die *.tra-Dateien, die man schon für die Installation braucht. Man kann alle folgenden Texte auch in *.tra-Dateien schreiben. Damit vom Programm auf sie zugegriffen werden kann, müssen sie hier schon erwähnt werden. Diese Zeile muss man auch für alle anderen Sprachen schreiben.

BEGIN ~Mod-Name~

Hiermit beginnt man die Installation.
Nun müssen die Dateien kopiert werden.

COPY ~char/.bmp~ ~override/*.bmp~*

für die beiden Bilder (wenn man neue benutzt). Danach kopiert man die Dateien des Charakters.

COPY ~char/char.bcs~ ~override/char.bcs~
COPY ~char/char.cre~ ~override/char.cre~

Jetzt muss man dem Charakter noch einen Namen geben; genauer gesagt zwei: einen kurzen Rufnamen und einen vollständigen Namen.

SAY NAME1 ~Rufname~
SAY NAME2 ~Vorname Zuname~

Alternativ kann man die beiden Namen auch durch *@1/@2* ersetzen und in eine setup.tra Datei schreiben (s.o.). Nun braucht ein Charakter noch ein Vorgeschichte bzw. eine Biographie. Hier kann man beliebig viel Aufwand treiben. Das von mir angeführte Beispiel ist allerdings zu kurz ;)
.

SAY BIO ~Char geboren, aufgewachsen, ein Held geworden.~

Jetzt müssen noch die Skripte einiger Gebiete ergänzt werden.

EXTEND_ BOTTOM ~Ar0900.bcs~ ~char/Archar.bcs~

Damit wird jetzt in dem Gebiet *Ar0900* unser Charakter erschaffen.

EXTEND_ TOP ~Ar4500.bcs~ ~char/char_ ein.bcs~

Damit ist das Einsprengsel auf den neuesten Stand gebracht worden (für nicht NPCs entfällt diese Zeile).
Als nächstes werden die Dialoge übersetzt.

COMPILE ~char/char.d~

Je nachdem, ob man *.tra-Dateien benutzt oder nicht ändert sich die nächste Zeile. ohne *.tra-Dateien:

USING ~~

mit *.tra-Dateien:

USING ~char/% s/char.tra~

Das *% s* steht für das Verzeichnis der jeweiligen Sprache, die bei der Installation später ausgewählt wird. Weiter mit

COMPILE ~char/charP.d~
USING ~~

und so zwar für alle *.d-Dateien, die angelegt wurden.
Jetzt gilt es noch, zwei weitere Dinge zu erledigen (nur für NPCs).

APPEND ~interdia.2da~ ~CHAR Bchar~

Dies fügt der Datei *interdia.2da* die untere Zeile hinzu. *CHAR* ist die DV und *Bchar* die Datei für die Dialoge mit anderen.

UNLESS ~CHAR~

Dieser Befehl sorgt dafür, dass die Zeile nur eingefügt wird, wenn es sie nicht schon gibt.

UNLESS ~25POST~

Dieser Befehl sorgt dafür, dass die Zeile nur eingefügt wird, wenn TdB nicht installiert ist.

APPEND ~interdia.2da~
~CHAR Bchar Bchar25~
UNLESS ~CHAR~
IF ~25POST~

Das ist jetzt dasselbe, wenn die Erweiterung installiert ist. Die Zeile wurde um die Dialog-Datei für TdB (*Bchar25*) ergänzt.

APPEND ~ pdialog.2da~
~CHAR charP charJ charD~
UNLESS ~CHAR~
UNLESS ~25POST~

Dies fügt eine Zeile in die Datei *pdialog.2da* ein. *CHAR* ist wieder die DV, *charP* und *charJ* sind die *.dlg-Dateien. *charD* ist ein Skript für Traumsequenzen. Wir haben hier keine benutzt, schreiben sie aber der Vollständigkeit halber hinein. Das gleich noch für die Erweiterung:

APPEND ~pdialog.2da~
~CHAR charP charJ charD char25P char25J char25D char25~
UNLESS ~CHAR~
IF ~25POST~

Hinzugekommen sind die Dialogdateien *char25P* und *char25J* sowie das Skript *char25D*, das für Traumsequenzen zuständig ist, und das Skript *char25*, das ein neues *Override script* für die Erweitertung ist. Wirklich nötig ist nur die *char25P*. Und schon sind wir mit diesem Punkt fertig.

7 Die *.zip-Datei

Jetzt müssen wir nur noch schauen, dass alle Dateien so in die *.zip-Datei eingebunden werden, dass sie das Installationsprogramm auch findet. Da nicht jeder WeiDU hat, legen wir auch noch eine Kopie der *WeiDU.exe*-Datei dazu. Diese benennen wir in *Setup-Char.exe* um. Wir nehmen uns ein Laufwerk und kopieren dort diese *.exe-Datei hin. Ebenfalls dazu kopieren wir die *.tp2-Datei, die *Setup-Char.tp2* genannt wird. Dazu kommt der Ordner *char*. In diesen Ordner kommt die

*.cre-Datei, die *.d-Dateien und die *.bcs-Dateien und die Bild-Dateien. Wenn wir *.tra-Dateien benutzen legen wir innerhalb des *char*-Ordners noch die Ordner für die jeweiligen Sprachen an und kopieren dort die *.tra-Dateien hinein. Dann legen wir in dem *char*-Ordner noch einen Ordner mit Namen *char-backup* an. Dort erstellen wir eine *Backup.txt*-Datei, deren Inhalt egal ist, aber lauten könnte: "Diese Datei wurde benötigt, um den Backup-Ordner an richtiger Stelle zu erstellen."
Wir öffnen nun das Programm WinZip und zwar nicht den Wizard sondern die Classic Variante. Wir erstellen ein neues Archiv auf dem Laufwerk, auf das wir die ganzen Dateien kopiert haben, der Name ist egal. Mittels *Add* kopieren wir alle Dateien in dieses Archiv; dabei ist darauf zu achten, das die Pfad-Information einer Datei erhalten bleibt. Letzteres erkennt man daran, dass im Archiv hinter den Dateien ein Pfad angeben ist. Für die *Setup-Char.** ist keine Pfadangabe nötig. Man darf auch nicht die *Backup.txt-Datei* vergessen (auch sonst keine)!
Diese *.zip-Datei muss dann im BG2-Verzeichnis entpackt werden. Danach muss mittels Doppelklick die Datei *Setup-Char.exe* gestartet werden. Wenn Sie alles richtig gemacht haben, ist dann Ihr Mod installiert.

8 Bemerkungen

Ich habe meistens die Dateien mit Char-irgendwas bezeichnet. Statt Char kann man jeden anderen Namen nehmen; es soll hier nur die Abkürzung für Charakter sein. Eigentlich sollte man, um Verwechslungen zu vermeiden auf jeden Fall einen anderen Namen wählen.
Da ich keine grosse Leuchte in diesen Dingen bin, habe ich mir alles mit Hilfe vieler anderer angeeignet. Diejenigen, die mir gerade einfallen sind: Dyara, Knight, Arian ("Tashia"-Mod), Greyfhain ("NPC-Tutorial"), Blue the immortal bard ("Banter-Tutorial") Wesley Weimer (viele Mods), NightShadows Forum und das Rosenranken-Forum. Danke, dass Ihr Euer Wissen mit mir geteilt habt!
Ein weiteres Dankeschön geht an die "Beta-Tester" Drake und Seradin.
Viel Spass!

P.S.: Bei Fragen o.ä. email an maus@fphfslinux.physik.uni-karlsruhe.de