Hallo Stalker,

existence hat S.T.A.L.K.E.R. leider den Rücken gekehrt um sich anderen Projekten zu widmen. Vielleicht wird er sich irgendwann wieder zu uns gesellen, aber bis dahin hab ich von ihm die Erlaubnis seine FAQ zu verändern und weiterzuführen. Als erstes werd ich hier für Ordnung und Übersicht sorgen. Dabei muß ich feststellen, daß das hier eigentlich keine FAQ ist sondern vielmehr ein Kompendium, also eine Wissensammlung handelt. So werde ich das auch fortführen.
Ich hab mir das so vorgestellt: Jeder Post präsentiert ein in sich geschlossenes Thema, wobei ein Post auch auf einen anderen Aufbauen kann. Leider sind die Posts im Forum chronologisch sortiert und ich kann sie selbst als Moderator nicht nach Themen umsortieren. Damit man trotzdem schnell findet was man sucht, füge ich hier eine Art Inhaltsverzeichnis mit Links zu den Posts ein.

Regeln:
Achtet auf die Richtigkeit der von euch geposteten Informationen. Probiert eure Tutorials am besten bei euch selbst aus, bevor ihr sie hier postet.
Es sind nur Toturials und Artikel erlaubt. Für Fragen und Kommentare steht euch der Thread Mod-FAQ: Fragen, Wünsche, Kommentare offen.
Mehrfachposts sind hier ausnahmsweise erlaubt, wenn es sich dabei um seperate Turorials oder Artikel handelt.
Achtet bitte auf eine akzeptable Rechtschreibung und eine übersichliche Gestaltung. Eure Leser sollen nicht schon an der (Un)Lesbarkeit eurer Beiträge scheitern
Seid sparsam mit Bildern und achtet darauf, daß die Dateigröße so klein wie möglich und so groß wie nötig ist. Wählt dazu ein geeignetes Bildformat und eine angemessene Kompression.
Die Bilder sollten nicht breiter als 640 Pixel sein, damit sie das Forumlayout nicht sprengen. Sollten große Bilder unbedingt nötig sein, dann verlinkt sie, am besten mit einem Thumbnail.
Ein Hinweis zu den Bildern, die ihr im Forum hochladet: Anscheinend werden diese Bilder nach einiger Zeit wieder entfernt. Ob und unter welchen Bedingungen das genau geschieht, weis ich leicder nicht. Ladet eure Bilder also besser auf euren Webspace oder mit einem Hostingservice hoch.

~ nihilant ~

Inhalt:

Allgemeines:
Nahezu alle Spieldaten wie Level, Texturen, Sounds usw sind in den Paketen gamedata.db0 bis .dbb zusammengefasst. Um sie einsehen und bearbeiten zu können, müssen diese Daten zuerst extrahiert werden. Das Tool, das ihr dazu benötig findet ihr hier:
download: Entpacker

Achtung! Die extrahierten Dateien sind zusammen 5,75 Gigabyte groß!

Achtung! Die gamedata.dbb wurde erst mit den Patches hinzugefügt und überschreibt einige Dateien aus den anderen Paketen. Entpackt dieses Paket in jedem Fall zuletzt!

Entpackt die Daten am besten in einen neuen Ordner. Wenn ihr alle db*-Paket entpackt habt, solltet ihr in eurem neuen Ordner folgendes sein (Ordner sind in eckigen Klammern[]):
[ai]
[anims]
[config]
[levels]
[meshes]
[scripts]
[shaders]
[sounds]
[spawns]
[textures]
game.graph
gamemtl.xr
lanims.xr
particles.xr
resource.h
senviroment.xr
shaders.xr
shaders_xrlc.xr
stalkergame.inf
Ich empfehle euch die Dateien, die ihr verändern wollt, in einen extra Arbeitsordner zu kopieren und die Originaldateinen so zu belassen, wie sie sind. Dann braucht ihr die Dateien im Fall eines Falles die Dateinen nicht nochmal zu entpacken.

Um eure modifizierten Dateien im Spiel zu benutzen müsst ihr nur einen Ordner namens gamedata im Stalkerverzeichnis erstellen und die Dateien mitsamt ihrer Ordnerstuktur dahinein verschieben oder kopieren.

Zum Beispiel die actor.ltx von
<euerordner>\config\creatures\actor.ltx
nach
...\gamedata\config\creatures\actor.ltx

_____

Anmerkung: existence's Originalpost war viel umfangreicher, aber auch sehr viel unübersichtlicher. Ich habe seinen Originalpost gespeichert und werde alles in neuen Posts nachreichen.

~ nihlant ~

Dialog-Tutorial

Das Dialogsystem von Stalker basiert auf XML, einer Auszeichnungssprache zur Darstellung

hirarchisch strukturierter Daten (wiki-Link). Dabei stehen die Dialoge für sich und sind nicht an NPCs gebunden. Ich will hier versuchen, das System anhand eines Tutorials zu erklären. Am besten schreibt man sich als erstes seinen Dialog mit allen Optionen auf ein Blatt Papier:

Spieler: Ich bin neu hier und kenn mich nicht so gut aus. Hast du ein paar Tipps für mich?
NPC: Wie wärs, wenn du erst mal die SuFu nutzt. Die Frage wurde schon tausend Mal beantwortet!
'-> Spieler: Hab ich gemacht, aber ich hab nichts gefunden.
| NPC: Dann such nochmal!
|
'-> Spieler: Ok, sorry. Werd ich gleich nachholen.
| NPC: So ists recht.
|
'-> Spieler: Hä?!
NPC: Na die Suchfunktion ... Ach egal. Laß dich einfach nicht von den Wildschweinen beissen.

Die Dateien mit den Dialogen befinden sich im Ordner gamedata\config\gameplay. Wir wollen unseren Dialog Wolf im Startdorf zufügen. Die Dialoge für Kordon stehen in der Datei dialogs_escape.xml.
Nochmal: Die Dialoge sind nicht an Charakter gebunden und stehen auch in anderen Dateien. In dialogs_escape.xml stehen nur Dialoge. Die Zuweisung zu einem NPC erfolgt später in einer anderen Datei.

Also ans Eingemachte:
Jede Dialogdatei fängt mit einem xml-Header an:

Code:
<?xml version="1.0" encoding="windows-1251" ?>Um den müssen wir uns aber nicht weiter kümmern. Ich erwähns hier nur der Vollständigkeit halber.
Danach folgt ein öffnender game_dialogs-Tag, der erst ganz am Ende der Datei wieder geschlossen wird. Alle Dialoge in einer Datei sind diesem game_dialogs-Element untergeordnet. In der nächsten Ebene folgen auch schon die Dialoge, die die Textschnipsel(Phrasen) beinhalten:

Code:
<dialog id="...">
  <phrase_list>
    <!-- der Dialog -->
  </phrase_list>
</dialog>Jedem dialog-Element muß ein id-Argument zugewiesen sein. Mit dieser ID weisen wir später den NPCs unseren Dialog zu. Für unseren Dialog wähle ich "test_smalltalk".
Die Dialoge bestehen aus Phrasen, jede mit einer Nummer. Die Startphrase ist immer die Phrase 0(Null). Wenn der Spieler den NPC grade erst anspricht, dann ist das text-Element leer. Ansonsten ist die Phrase 0 der Satz mit dem der Spieler den Dialog in einem Gespräch startet. Spieler und NPC wechseln sich immer ab. In den Phrasen können next-Elemente stehen, die angeben mit welcher Phrase es weiter geht oder welche Auswahlmöglichkeiten der Spieler hat. Hat eine Phrase kein next-Element, dann ist der Dialog beendet und es geht mit dem nächsten Dialog des NPCs weiter, oder das Gespräch ist ganz beendet. Unser Dialog soll mit einer Frage wärend eines Gesprächs beginnen.

Code:
<dialog id="test_smalltalk">
  <phrase_list>
    <phrase id="0">

```
      <text>Ich bin neu hier und kenn mich nicht so gut aus. Hast du ein paar Tipps für mich?
</text>
      <next>1</next>
    </phrase>
    <phrase id="1">
      <text>Wie wärs, wenn du erst mal die SuFu nutzt. Die Frage wurde schon tausend Mal
beantwortet!</text>
      <next>11</next>
      <next>12</next>
      <next>13</next>
      </phrase>
```
...die Phrase 1 hat drei next-elemente: das sind die Auswahlmöglichkeiten des Spielers.
Code:
```
...
    <phrase id="11">
      <text>Hab ich gemacht, aber ich hab nichts gefunden.</text>
      <next>111</next>
    </phrase>
    <phrase id="111">
      <text>Dann such nochmal!</text>
    </phrase>
    <phrase id="12">
      <text>Ok, sorry. Werd ich gleich nachholen.</text>
      <next>121</next>
    </phrase>
    <phrase id="121">
      <text>So ists recht.</text>
    </phrase>
    <phrase id="13">
      <text>Hä?!</text>
      <next>131</next>
    </phrase>
    <phrase id="131">
      <text>Na die Suchfunktion ... Ach egal. Laß dich einfach nicht von den Wildschweinen
beissen.</text>
    </phrase>
  </phrase_list>
</dialog>
```
Damit wäre unser Dialog auch schon fertig. Die Phrasen 111, 121 und 131 haben keine weiteren next-Einträge. Der Dialog ist damit beendet. Hat der NPC aber noch weitere Dialoge, geht es mit denen weiter. Das sehen wir aber später.

Wenn ihr euch jetzt aber die bereits vorhandenen Dialoge in den Spieldateien anseht, wird euch auffallen, daß statt Texten kryptische Bezeichner in den Text-Elementen stehen. Das sind Platzhalter für die eigenlichen Texte, die erst im Spiel geladen werden. Auf diese Weise können für jede Sprache andere Texte geladen werden.
Beim Erstellen eines Dialogs ist es zweckmässig, die Texte zuerst direkt einzutragen, damit es übersichtlicher ist. Wenn der Dialog dann so funktioniert wie man es haben möchte, dann kann man die Texte immer noch in die Lokalisationsdateien auslagern. Das sind übrigens auch XML-Dateinen und befinden sich für die deutsche Version im Ordner gamedata\config\text\ger.
Ich mach das hier mal vor:

Code:

```
<dialog id="test_smalltalk">
  <phrase_list>
    <phrase id="0">
      <text>test_smalltalk_0</text>
      <next>1</next>
    </phrase>
    <phrase id="1">
      <text>test_smalltalk_1</text>
      <next>11</next>
      <next>12</next>
      <next>13</next>
    </phrase>
    <phrase id="11">
      <text>test_smalltalk_11</text>
      <next>111</next>
    </phrase>
    <phrase id="111">
      <text>test_smalltalk_111</text>
    </phrase>
    <phrase id="12">
      <text>test_smalltalk_12</text>
      <next>121</next>
    </phrase>
    <phrase id="121">
      <text>test_smalltalk_121</text>
    </phrase>
   <phrase id="13">
      <text>test_smalltalk_13</text>
      <next>131</next>
    </phrase>
    <phrase id="131">
      <text>test_smalltalk_131</text>
    </phrase>
  </phrase_list>
</dialog>in die Datei gamedata\config\text\ger\stable_dialogs_escape.xml kommt dann folgendes:

Code:
  <string id="test_smalltalk_0">
    <text>Ich bin neu hier und kenn mich nicht so gut aus. Hast du ein paar Tipps für mich?</text>
  </string>
  <string id="test_smalltalk_1">
    <text>Wie wärs, wenn du erst mal die SuFu nutzt. Die Frage wurde schon tausend Mal
beantwortet!</text>
  </string>
  <string id="test_smalltalk_11">
    <text>Hab ich gemacht, aber ich hab nichts gefunden.</text>
  </string>
  <string id="test_smalltalk_111">
    <text>Dann such nochmal!</text>
  </string>
  <string id="test_smalltalk_12">
    <text>Ok, sorry. Werd ich gleich nachholen.</text>
```

```
  </string>
  <string id="test_smalltalk_121">
    <text>So ists recht.</text>
  </string>
  <string id="test_smalltalk_13">
    <text>Hä?!</text>
  </string>
  <string id="test_smalltalk_131">
    <text>Na die Suchfunktion ... Ach egal. Laß dich einfach nicht von den Wildschweinen
beissen.</text>
  </string>Wollen wir unseren Dialog auch im Englischen verfügbar machen, dann müssen wir nur
die Einträge für die Text-IDs im text\eng-Ordner einfügen und übersetzen. Lassen wir das weg,
sehen die Spieler auf Englisch nur die Text-IDs, mit denen sie natürlich nichts anfangen können.

Jetzt müssen wir nur noch unseren Dialog einem NPC zuweisen. Die Beschreibungen der NPCs
sind, haltet euch fest, ebenfalls XML-Dateinen und befinden sich im selben Ordner wie die Dialoge.
Das hier ist ein Auszug aus der Datei character_desc_escape.xml, der Wolf beschreibt:

Code:
<specific_character id="esc_wolf" team_default = "1">
  <name>esc_wolf_name</name>
  <icon>ui_npc_u_stalker_neytral_balon_1</icon>
  <bio>esc_wolf_bio</bio>

  <class>esc_wolf</class>
  <community>stalker</community>
  <terrain_sect>stalker_terrain</terrain_sect>

  <rank>434</rank>
  <reputation>5</reputation>
  <money min="600" max="2000" infinitive="0"/>

  <snd_config>characters_voice\human_01\stalker\</snd_config>
  <crouch_type>-1</crouch_type>
  <visual>actors\neytral\stalker_neytral_balon_1</visual>

  <supplies>
    [spawn] \n
    wpn_pm \n
    ammo_9x18_fmj = 1 \n
    wpn_ak74u \n
    ammo_5.45x39_fmj \n
    device_torch \n
    hand_radio \n
  </supplies>

#include "gameplay\character_criticals_4.xml"

  <start_dialog>escape_lager_volk_talk</start_dialog>
  <start_dialog>dm_hello_dialog</start_dialog>
  <actor_dialog>dm_cool_info_dialog</actor_dialog>
  <actor_dialog>dm_help_wounded_medkit_dialog</actor_dialog>
```

<actor_dialog>tm_wolf_dialog</actor_dialog>
<actor_dialog>tm_wolf_reward</actor_dialog>
</specific_character>Uns interessiern hier nur die letzten Zeilen und wie wir sehen können, hat Wolf schon einige Dialoge. Wir fügen unseren einfach in ein neues actor_dialog-Element ein:

Code:
...
<start_dialog>escape_lager_volk_talk</start_dialog>
<start_dialog>dm_hello_dialog</start_dialog>
<actor_dialog>dm_cool_info_dialog</actor_dialog>
<actor_dialog>dm_help_wounded_medkit_dialog</actor_dialog>
<actor_dialog>tm_wolf_dialog</actor_dialog>
<actor_dialog>tm_wolf_reward</actor_dialog>
<actor_dialog>test_smalltalk</actor_dialog>
</specific_character>Das wars! Wenn man Wolf jetzt anspricht, hat man eine neue Dialogoption, mit der man unseren Dialog startet.(siehe angehängten Screenshot)

Die geänderten Dateien aus dem Tutorial könnt ihr euch hier runterladen:
http://nsite.de/public/dialog_tutorial.rar

Erstellung eines eigenen Fadenkreuzes

Erstellung eines eigenen Fadenkreuzes
_____von existence

Die Grafik-Dateien liegen im dds-Format vor.
Dieses Format kann man leider nicht mit gängigen Programmen bearbeiten.
Eine kostenlose und sehr effiziente Alternative zu Photoshop stellt "GIMP" dar.
Ein Plug-in für dds-Formate ist ebenfalls kostenlos zu erhalten.
Um Icons zu erstellen oder Skins zu bearbeiten, steht GIMP Photoshop kaum nach.
http://the-gimp.softonic.de

Alle Bilder sind übrigens auch mit GIMP erstellt.

In den noch kommenden Schritten werde ich das Tool auch noch näher beschreiben, damit man langsam die Anfänger-Ebene verlassen kann. Das Fadenkreuz ist aber noch recht simpel.

Das Zielkreuz vom G36 ist heute mal fällig.
Bei Arbeiten wie diesen ist künstlerische Freiheit gefragt, oder die Nähe zur Realität. Daher vermittle ich nur die Technik für die Erstellung.

Die nötige Datei ist
gamedata\textures\wpn\wpn_crosshair_g36.dds
In GIMP laden, loslegen.
Der aufmerksame User wird bemerken, dass wir es plötzlich mit einem Ei zu tun haben.

Wer sich sicher fühlt, kann auch gerne jetzt schon munter loslegen
Ich empfehle jedoch, das Ding rund zu machen.

Dazu klickt Ihr auf Bild -> Bild skalieren. Die Verlinkung von x und y aufheben, Prozent wählen und für die Breite (x) 134% eintragen. Das war´s auch schon.

Bevor wir irgendwas verändern, suchen wir uns nochmal den Mittelpunkt raus.
Dazu auf Bild -> Hilfslinie -> neue Hilflinie... klicken. Richtung ist horizontal mit der Position 512.
Danach gleich noch eine. Vertikal mit 686.
Offensichtlich ist der Mittelpunkt gut getroffen. Nun kann geschmiert werden.

Vor dem Abspeichern daran denken, dass das Bild wieder in das ursprüngliche Format zurückgesetzt werden muss. Die Abmessung ist 1024x1024.
Viel Spass. Backup nicht vergessen!

Zwar sind schon genug Zielkreuze im Umlauf, aber mein eigenes sieht so aus:

Natürlich ist es vieeeeeeeeeeeeel näher am Original

Icons verändern

Icons verändern
_____von existence

Wichtig

Um unschöne Effekte zu vermeiden, sollten die dds-Dateien ohne Kompression abgespeichert werden (DXT3 geht).

Als Einstieg wähle ich die gebräuchlichsten Icons, die auch vermehrt in Mods Verwendung finden. Die Icons von Waffen, Artefakten und anderen Items, die man im Inventar sieht.

Parameter wie diese

inv_grid_width = 5
inv_grid_height = 2
inv_grid_x = 5
inv_grid_y = 34

verweisen im Falle der Icons, die im Inventar sichtbar sind auf
\gamedata\textures\ui\ui_icon_equipment.dds.

Wie oben beschrieben, geben die Koordinaten den Ort an, da nicht für jedes Icon eine Grafik vorliegt, sondern sie sich in einer grossen "Map" befinden. Es läuft wie beim Schiffeversenken. Richtige Koordinate, richtiges Item. Um die Sache nicht zum Ratespiel werden zu lassen, gebe ich ein paar Beispiele.

So sieht die Datei aus:
http://s6.bilder-hosting.de/img/YERA2.jpg(371x742, 194kb)

Sieht zunächst alles etwas unsortiert aus, wird aber sehr schnell übersichtlich.
Erstellt eine weitere Ebene in der Grösse 1024x2048.
Die Ebenenfüllart auf "Transparenz" stellen.
Diese zunächst unsichtbare Ebene wird im Ebenenmanager unter die "Main Surface" geklickt.

Danach "Filter -> Render -> Muster -> Schachbrett" wählen. Als Grösse für das Muster 50 Pixel eingeben. Das fertige Produkt sollte dann in etwa so aussehen:
http://s6.bilder-hosting.de/img/YF3EX.jpg(371x742, 246kb)

Taaataaa. Es ist Ordnung im Chaos und es lassen sich nun die Koordinaten der Icons ermitteln.
Der Koordinatenursprung ist links oben. Also 0,0.
Um beispielsweise das Icon des Dragunov SVD zu erhalten, gibt man folgenden Wert an:

inv_grid_width = 6
inv_grid_height = 2
inv_grid_x = 0
inv_grid_y = 0

Damit wird ein virtueller Kasten erstellt. Immer merken, dass die linke, obere Ecke der "Anfang" des Kastens ist. Er beginnt also auf x = 0, y = 0, hat eine Länge von sechs Kästchen (x) und eine Höhe (hier ist es eigentlich Tiefe, haha) von zwei Kästchen.

Schön, aber wie erstellt man nun eigene Icons auf der Map?
Ungefähr so...
Wählt im Ebenenmanager die "Main Surface". Danach klickt bei den Werkzeugen die rechteckige Auswahl an und teilt ihr eine feste Grösse zu. Hier macht es Sinn, sich an das Raster zu halten. Ein Kästchen hat 50 Pixel, also braucht man für das Dragunov eine Auswahlgrösse von 300x100 Pixel. Diese Auswahl wendet man nun passend auf das Gewehr an.
Nun einen Rechtsklick machen und kopieren. Nun lässt sich das Gewehr ebenfalls bewegen, es ist eine "schwebende Auswahl". Mit einem Linksklick kann man es an beliebiger Stelle wieder in der Ebene verankern.
Sieht so aus:
http://s6.bilder-hosting.de/img/YFO3O.jpg(512x384, 102kb)

Okay, aber ich habe eine andere Geschichte vor. Wir löschen das Icon!
Dazu einen Rechtsklick -> Ebene -> Ebene löschen. Weg ist es. Aber was machen wir mit der Lücke? Nun...wir klauen uns irgendwo etwas und stopfen die Lücke.
Ich habe mir ein Bild von einem PSG-1 gesucht. Wer es nicht kennt, der sieht es jetzt

Noch hat es einen sichtbaren Hintergrund und "guckt" in die falsche Richtung. Kriegen wir aber alles hin.
Dieses Bild öffnen wir nun zusätzlich mit GIMP.
Schnell auf Bild -> Transformation -> Horizontal spiegeln klicken.
Ja, wir haben nun eine optimale Waffe für Linkshänder geschaffen. Bundis, Ruhe bewahren, ich weiss, was ich tue.

Wir haben einen tollen Hintergrund, daher kommt der allseits beliebte Zauberstab nun voll zur Geltung. Anwählen, den weissen Hintergrund anklicken und sich freuen, dass es nun so aussieht:

Diese Auswahl wird nun einfach kopiert und wir wechseln wieder zu unserer Icon-Map.

Dort wird Geschichte mit einem Rechtsklick wie gewohnt eingefügt.

http://s6.bilder-hosting.de/img/YGTSQ.jpg(512x384, 114kb)

Noch ist das Gewehr zu gross, also passen wir es an.
Ebene -> Ebene skalieren. Ein Dialog öffnet sich. Wählt als Einheit Prozent und achtet darauf, dass x und y verlinkt sind (sollte aber der Fall sein). Gebt 60% ein und schiebt das neue Icon in die richtige Position.

Wenn wir nun sauber gearbeitet haben, dürfte das neue Icon am rechten Platz sein und richtig gut aussehen:

Die Ebene mit dem Schachbrett kann nun wieder entfernt werden und wir speichern die modifizierte Datei unter gamedata\textures\ui\ui_icon_equipment.dds ab.
Vorher natürlich ein Backup vom Original machen.
Beim Speichern auf überflüssige Kompression verzichten.

Wer weiter oben schon gelernt hat, wie man die Beschreibungen der Waffen verändert, wird dann im Spiel sowas erleben können:
http://s6.bilder-hosting.de/img/YINVF.jpg(747x560, 282kb)

Leider wird es erst mit dem SDK die Möglichkeit geben, auch ein neues Modell einzubinden.

_____
Anmerkung: Ich habe zu große Bilder nur verlinkt. Bild- und Dateigrößen stehen hinter den Links. Man muß schließlich auch daran denken, daß hier bald noch mehr Artikel mit noch mehr Bildern stehen. Mehrere MB pro Seite sind in einem Forum einfach unangemessen.

Wie funktionieren Waffen in Stalker
_____von nihilant und Krishty
Die Waffen in Stalker bestehen aus folgenden Elementen:
Modelle und Animationen
Für jede Waffe werden zwei Modelle benutzt: Eines für die Ego-Perspektive und eine für die Aussenansicht. Die Animationen sind in die Modelldateien integriert. Das Dateiformat ist OGF, ein GSC-hauseigenes Format, das bisher leider nicht von Moddern bearbeitet werden konnte.
Texturen
Beide Modelle verwenden die selbe Textur. Der Textur ist in der textures.ltx eventuell noch eine Bumpmap zugewiesen, das ist aber nicht zwingend notwendig. Das Dateiformat ist .dds (Direct Draw Surface).
Icon
Alle Icons befinden sich in der selben Textur: gamedata\textures\ui\ui_icon_equipment.dds. In der Konfigurationsdatei steht welcher Bereich in der Textur das Icon für die Waffe darstellt.
Sounds
Die Geräusche, die die Waffe beim Schießen, Nachladen, beim Ziehen usw macht. Das Dateiformat ist OGG, der Codec ist Vorbis. Die Sounds müssen in 16bit, 44kHz und in Mono codiert sein, damit die Xray-Engine sie lesen kann. Die Engine sucht in den Sounddateien auch nach einem speziellen Kommentar, in dem Parameter wie Lautstärke, KI-Reichweite usw. stehen. Das Spiel läuft auch ohne diese Kommentare, es hat aber Vorteile den Kommentar trotzdem einzufügen. Weiteres zum Thema findet ihr hier: der missing ogg-comment error.

Eine C++-Klasse in der Spieleengine
Damit die Engine überhaupt weiß, was es mit den ganzen Daten anfangen soll, muss darin eine Waffenklasse(Programmcode) implementiert sein. Die Engine beinhaltet für bestimmte Waffenarten bereits einige Klassen, beispielsweise für Sturmgewehre oder Schrotgewehre. Jede Klasse hat andere Fähigkeiten, so beherrscht nur die Sturmgewehrklasse Feuermodi wie Feuerstoß oder Dauerfeuer. Die Klassen sind bereits in Maschinensprache kompiliert und wir können sie nicht verändern. Wir können einer Waffe nur eine passende Klasse zuweisen.
Konfiguration
Das Herzstück der Waffen. Die Konfigurationsdateien sind ltx-Dateien und befinden sich im Ordner gamedata\config\weapons. Da sie in Klartext vorliegen, können wir sie leicht modifizieren. Sie faßt alle Informationen zusammen: Welche Modelle sollen verwendet werden, welche Animationen, Sounds, welche Munition soll benutzt werden können, welche Feuermodi sollen verfügbar sein und so weiter und so fort.
Namen- und Beschreibungstexte
Um die Lokalisierung zu erleichtern, stehen die Namen und Beschreibungen nicht direkt in den Konfigurationen, sondern sind in weitere Dateien ausgelagert. Sie befinden sich in den Ordnern gamedata\config\text\<Sprachkürzel> und sind im xml-Format verfasst. In den Konfigurationen stehen Platzhalter, die sich auch in den xml-Dateien wiederfinden. Die Engine wählt je nach ausgewählter Sprache den passenden String aus diesen Dateien.

Allgemeines zu den Konfigurationsdateien

Die ltx-Dateien funktionieren wie INI-Dateien: Sie enthalten Sektionen mit Elementen, denen Werte zugewiesen können (aber nicht müssen). Sektionen müssen einen einzigartigen Namen haben und stehen in eckigen Klammern, zB. [meine_sektion]. Sektionen können von anderen Sektionen abgeleitet werden. Das bedeutet, daß sie alle Eigenschaften von der Vatersektion erben. Die abgeleitete Sektion kann diese Eigenschaften aber überschreiben. Ein Beispiel:
[meine_erste_sektion]
eigenschaft1 = irgendeineVariabel
eigenschaft2 = "einString"
eigenschaft3 = 1337 ;eine Zahl

[meine_zweite_sektion]:meine_erste_sektion
eigenschaft3 = 9999
eigenschaft4 = 0.223, 0.144, 0.514; eine Eigenschaft kann auch mehrere Werte haben.
Die Sektion meine_zweite_sektion hat vier Eigenschaften, drei davon hat sie von meine_erste_sektion geerbt und eine davon überschrieben. Nützlich ist das wenn man viele ähnliche Sektionen erstellen will - man kann sich eine menge Schreibarbeit sparen. Die einzigartigen Waffen sind von den Standartwaffen abgeleitet. Sie erben alle Eigenschaften der normlaen Waffen und überschreiben nur die Eigenschaften die verändert werden sollen.

Die Waffenkonfigs

Wie oben beschrieben stehen in den Waffenkonfigurationen alle Informationen zur Waffe. Sie ist das Bindeglied zwischen den ganzen Modellen, Sounds usw. Ich muß zugeben, daß ich nicht die Bedeutung von allen Einträgen kenne. Viele Eigenschaften scheinen auch garnicht mehr vom Spiel geladen zu werden und sind Überbleibsel aus früheren Versionen des Spiels. Ich liste hier alle Einträge auf die ich kenne und versuche sie so gut zu erklären wie ich kann. Wer Ergänzungen, Berichtigungen oder sowas hat, kann sicher gerne per PN an mich wenden.

GroupControlSection - Immer spawn_group.

discovery_dependency - ??? Immer leer.
$spawn - Name im Leveleditor.
$prefetch - Prefetch-Einstellung. Prefetch-Beschreibungen lesen.
class - Die Klasse in der xrGame.dll, an welche diese Waffe gebunden wird. Sie entscheidet u.a. darüber, ob nur Einzelschuss geschossen werden kann (Pistolenklasse) oder ob Granatwerfer, Scope oder Silencer montiert werden können (dies ist allerdings auch vom Model abhängig).
cform - ??? Wahrscheinlich für Spielphysik. Immer skeleton.
launch_speed - ??? Wahrscheinlich unbenutzt und Schlacke aus einer früheren Engine-Version.
visual - Der Pfad zum Model (also der ogf-Datei) der Waffe im 3rd Person View.
ph_mass - Das Gewicht der Waffe, in kg. Ausschließlich für Spielphysik, hat keinen Einfluss auf das Inventar (dafür gibt es inv_weight).
normal - Normalenvektor für die Darstellung der Waffe im 3rd-Person-View. Immer (oder meistens) 0.0, 1.0, 0.0
position - Position der Waffe im 3rd Person View.
orientation - Rotation der Waffe im 3rd Person View, in Grad. Immer 0.0, 0.0, 0.0.
startup_ammo - ??? Wahrscheinlich unbenutzt und Schlacke aus einer früheren Engine-Version.

Inventaroptionen
slot - Der Slot, in den die Waffe passt:
0 - Messer
1 - Pistole
2 - Gewehr
animation_slot - Welche Animation im 3rd Person View für die Waffe ausgewählt wird (genau wie Slot). Ist dies z.B. der Pistolenslot, wedeln und kreisen die NPCs bei Langeweile mit der Waffe wie ein Cowboy mit seiner Pistole.
ef_main_weapon_type - ??? Multiplayer?
ef_weapon_type - ??? Multiplayer?
weapon_class - ??? KI?
inv_name - Name des Strings in der string_table_enc_weapons.xml, der den Namen der Waffe enthält. Wenn nicht vorhanden, wird der String direkt verwendet. Wird bspw. angezeigt, wenn man eine Waffe aufhebt oder wenn man sich im Inventar ihre Details ansieht.
inv_name_short - Name des Strings in der string_table_enc_weapons.xml, der den abgekürzten Namen der Waffe enthält. Wenn nicht vorhanden, wird der String direkt verwendet. Wird bspw. rechts unten im HUD angezeigt, wenn man die Waffe in der Hand hält.
description - Name des Strings in der string_table_enc_weapons.xml, der die Waffenbeschreibung liefert.
cost - Preis, in Rubel. Variiert je nach Händler, Kauf- und Verkaufumständen.
inv_weight - Leergewicht, in kg. Wird ausschließlich für das Inventar genutzt - für physikalische Berechnung gibt es ph_mass.
inv_grid_width - Siehe Icon-Beschreibungen.
inv_grid_height - "
inv_grid_x - "
inv_grid_y - "
kill_msg_x - ??? Multiplayer?
kill_msg_y - "
kill_msg_width - "
kill_msg_height - "
Player-Optionen
hand_dependence - ??? Wahrscheinlich unbenutzt und Schlacke aus einer früheren Engine-Version.
single_handed - Ob die Waffe beidhändig getragen wird (0) oder nur mit einer Hand (1). Selbst wenn die Waffe nur einhändig getragen wird, wie z.B. Pistolen, benutzt der Schütze die andere Hand, um die erste zu stützen.

holder_range_modifier - Wieviel weiter die Waffe schießt, wenn man sie angelegt abfeuert. Beispiel: Bei 2.0 schießt sie doppelt so weit wie in fire_distance angegeben. Die Flugbahn wird dadurch aber nicht gestreckt.
holder_fov_modifier - Faktor für die Größe des Sichtbereiches, wenn man die Waffe anlegt. Bespiel: Bei 0.5 ist das Sichtfeld doppelt so eng, also 2x Zoom.
zoom_enabled - Ob man mit der Waffe zoomen kann, indem man die rechte Maustaste benutzt.
zoom_hide_crosshair - Wahrscheinlich unbenutzt und Schlacke aus einer früheren Engine-Version.
control_inertion_factor - Um wieviel träger die Maus beim tragen der Waffe wird. Beispiel: 2.0 bedeutet, dass man die Maus doppelt so stark bewegen muss, um die Waffe zu schwenken.
sprint_allowed - Ob der Spieler mit gezogener Waffe sprinten kann (true) oder nicht (false).
use_aim_bullet - Wenn diese Option aktiviert ist, man die Waffe anlegt und damit lange genug zielt ohne die Waffe zu bewegen, wird die erste Kugel, die man abfeuert, punktgenau ins anvisierte Ziel schießen. Siehe auch...
time_to_aim - Die Zeit, in Sekunden, über die man die Waffe beim angelegten Zielen nicht bewegen darf, bis use_aim_bullet wirken kann. Nach dem Schuss muss man erneut über diese Zeitspanne warten und still zielen, bis es wirkt.
PDM_disp_base - Um wieviel die Waffe ungenauer wird wenn man sie aus der Hüfte - also ohne mit der rechten Maustaste zu zoomen - abfeuert.
PDM_disp_vel_factor - Skaliert PDM_disp_base beim Schwenken. Beispiel: Bei 2.0 wird die Waffe beim Schwenken doppelt so ungenau wie beim stillhalten.
PDM_disp_accel_factor - Skaliert PDM_disp_base mit der Laufgeschwindigkeit des Spielers.
PDM_crouch - ???
PDM_crouch_no_acc - ???
weapon_remove_time - ??? Nur bei sehr wenigen Waffen gefunden. Wahrscheinlich unbenutzt und Schlacke aus einer früheren Engine-Version.

NPC-Optionen
$npc - Sollen NPCs die Waffe benutzen können?
scheduled - ??? ALife?
min_radius - Minimale Distanz zum Ziel, in Meter, ab der NPCs die Waffe benutzen sollen
max_radius - Maximale Distanz zum Ziel, in Meter, bis zu der NPCs die Waffe benutzen sollen

Umhängen bzw. Wegstecken der Waffe
strap_position - Position der Waffe relativ zum Animation Bone.
strap_orientation - Rotation der Waffe, in Grad, relativ zum Animation Bone.
strap_bone0 - Animation Bone.
strap_bone1 - Weiterer Animation Bone.

Kinetischer Teil
ammo_class - Liste der Munitionstypen, die mit der Waffe kompatibel sind.
tracers - Ob die verschossenen Kugeln Leuchtspuren besitzen oder nicht. Überschreibt den Wert tracer in der Munitions-Section!
tracers_color_ID - Der Farbindex der verschossenen Kugeln. Überschreibt den Wert tracer_color_ID in der Munitions-Section! Siehe dort für Details.
ammo_limit - Ungenutzt. Wahrscheinlich Schlacke aus einer früheren Engine-Version.
ammo_current - Ungenutzt. Wahrscheinlich Schlacke aus einer früheren Engine-Version.
ammo_elapsed - Ungenutzt. Wahrscheinlich Schlacke aus einer früheren Engine-Version.
ammo_mag_size - Anzahl der Patronen, die in ein Magazin passen.
fire_modes - Liste aller Feuermodi:
-1 - Automatik
1 - Einzelschuss
2 - Doppelschuss

3 - Dreifachschuss [...]
direction - Die Schussrichtung der Waffe, als 3D-Vektor. Immer 0.0, 0.0, 1.0.
fire_dispersion_base - Die Streuung der Waffe, wenn man sie in ein Gestell spannt und abfeuert, in Grad. 0.0 ist der Idealwert, 1.0 bedeutet, dass die Schüsse einen Grad streuen. Dieser Wert wird außerdem mit k_disp in der momentanen Munitionsart multipliziert sowie mit den PDM_disp_XXXs (nur falls man nicht angelegt zielt) und...
fire_dispersion_condition_factor - Wie stark die Streuung vom Zustand der Waffe abhängt. Beispiel: 5.0 bedeutet, dass die verschmutzte Waffe fünf mal so stark streut wie die saubere.
misfire_probability - Wahrscheinlichkeit auf Ladehemmung bei einer perfekt geputzten Waffe. Beispiel: 0.001 bedeutet einer aus tausend Schuss. Kann natürlich früher kommen, ist ja zufällig.
misfire_condition_k - Wie stark die Wahrscheinlichkeit auf Ladehemmung durch fortschreitende Verschmutzung steigt. Realistisch scheinen Werte zwischen 0.05 (selten) und 0.15 (oft) zu sein.
bullet_speed - Die Mündungsgeschwindigkeit des Projektils, in m/s.
fire_distance - Entfernung, bis zu welcher die Kugeln verfolgt werden, in m. Hat keinen Einfluss auf die Schussbahn!
cam_relax_speed - Wie schnell der Schütze nach dem Schuss den Rückstoß ausgleicht, in 1/s. Nur bei Einzelfeuer und für die KI genutzt.
cam_relax_speed_ai - Wahrscheinlich unbenutzt (in cam_relax_speed übernommen worden) und Schlacke aus einer früheren Engine-Version.
cam_dispersion - Wie stark der Schütze nach dem Schuss verreißt, in Grad.
cam_dispersion_inc - ??? Wahrscheinlich, wieviel der Rückstoß von Schuss zu Schuss stärker wird.
cam_dispertion_frac - Wert zwischen 0 und 1, wie stark sich der Rückschlag nach oben auswirkt (wird nach diesem Wert gerichtet zufällig ausgewählt).
cam_step_angle_horz - Wie stark der Schütze nach dem Schuss nach links oder rechts verreißt, in Grad.
cam_max_angle - Maximaler Verriss des Schützen nach oben, in Grad. Beim Erreichen verliert die Waffe ihren Rückstoß.
cam_max_angle_horz - Maximaler Verriss des Schützen nach links und rechts, in Grad.
hit_type - Verletzungstyp: Immer Schussverletzung (fire_wound).
hit_power - Stärke der Verletzung bei Treffer. Wird mit k_hit in der momentanen Munitionsart multipliziert und ist von noch vielen anderen Faktoren abhängig.
hit_impulse - Impulsübertragung bei Treffer, in Kilonewton. Wird mit k_impulse in der momentanen Munitionsart multipliziert und ist von noch vielen anderen Faktoren abhängig.
condition_shot_dec - Wie stark der Zustand der Waffe sich mit jedem Schuss verschlechtert. Beispiel: 0.01 bedeutet, dass die Waffe nach 1 / 0.01 = 100 Schuss unbrauchbar ist. Wird mit impair in der momentanen Munitionsart multipliziert.
rpm - Schuss pro Minute.
rpm_empty_click - Wie oft soll das "Klick" wiederholt werden, wenn die Waffe leer ist?
flame_particles - Name des Partikeleffekts, der durch einen Schuss aktiviert wird und das Mündungsfeuer darstellt.
smoke_particles - Name des Partikeleffekts, der durch einen Schuss aktiviert wird und den Rauch des Mündungsfeuers darstellt.
fire_point - Koordinaten des Austrittspunkts des Schusses zum Abspielen der Effekte.
fire_point2 - Wie fire_point. Bei manchen Waffen kommen sogar noch mehr vor, scheint von der class abhängig zu sein.
shell_point - Koordinaten des Austrittspunkts der Hülsen.
shell_dir - Richtung, in welche die Hülsen wegfliegen, als 3D-Vektor.
shell_particles - Name des Partikeleffekts, der durch einen Schuss aktiviert wird und die Hülse darstellt.
light_disabled - Kein Mündungsfeuer (true) oder doch (false)? Wird oft bei Waffen mit Schalldämpfern genutzt.
light_color - Farbe des Lichts, in RGB je zwischen 0.0 und 1.0.

light_range - Reichweite des Lichts, in Metern.
light_var_color - ???
light_var_range - ???
light_time - Zeit in Sekunden, wie lange das Flackern dauert.
Sounds - Name, ggf. Prozent Lautstärke, ggf. Dauer in Sekunden
snd_draw - Sound, der beim Wegstecken der Waffe abgespielt wird.
snd_holster - Sound, der beim Anlegen der Waffe abgespielt wird.
snd_shoot - Name des Sounds, der bei einem Schuss abgespielt wird.
snd_empty - Sound, der beim versuchten Schuss aus einer leeren Waffe abgespielt wird.
snd_reload - Sound, der beim Nachladen abgespielt wird.
snd_switch - Sound, der beim Ändern des Feuermodus abgespielt wird.
Schalldämpfer - muss von class und model unterstützt werden! Kann auch ohne funktionieren und erst viel später im Spiel abstürzen!
silencer_status - Kann ein Schalldämpfer montiert werden?
0 - keiner
1 - fest
2 - optional
silencer_name - Name des Schalldämpfers. Dieser muss als Section dieses Namens vorliegen.
silencer_x - X-Position des Schalldämpfers auf dem Inventar-Icon der Waffe.
silencer_y - Y-"""
silencer_bullet_speed - Die Mündungsgeschwindigkeit des Projektils, in m/s. Typischerweise ein paar m/s schneller als ohne Siler.
silencer_fire_distance - Entfernung, bis zu welcher die Kugeln verfolgtwerden, in m. Siehe fire_distance.
silencer_hit_power - Stärke der Verletzung bei Treffer. Typischerweise minimal höher als ohne Siler. Siehe hit_power.
silencer_hit_impulse - Impulsübertragung bei Treffer, in kN. Typischerweise minimal höher als ohne Siler. Siehe hit_impulse.
Partikel- Licht und Soundeffekte (snd_silncer_shot): Siehe ohne Präfix silencer_.
Zielfernrohr - muss von class und model unterstützt werden! Kann auch ohne funktionieren und erst viel später im Spiel abstürzen!
scope_status - Kann ein Zielfernrohr montiert werden?
0 - keins
1 - fest
2 - optional
scope_name - Name des Zielfernrohrs. Dieses muss als Section dieses Namens vorliegen.
scope_x - X-Position des Zielfernrohr-Symbols auf dem Waffensymbol im Inventar.
scope_y - Y-"""
scope_zoom_factor - Zoomfaktor des Zielfernrohrs. Einheit unbekannt aber 8 dürfte 4x-Zoom entsprechen. WICHTIG: Die Variable muss auch initialisiert werden, wenn kein Scope genutzt wird (wird von einigen classes gefordert!).
scope_texture - Name der Textur des Zielfernrohrs, die erscheint, wenn man die Waffe anlegt.
Granatwerfer - muss von class und model unterstützt werden! Kann auch ohne funktionieren und erst viel später im Spiel abstürzen!
grenade_launcher_status - Kann ein Granatwerfer montiert werden?
0 - keiner
1 - fest
2 - optional
grenade_launcher_name - Name des Granatwerfers. Dieser muss als Section dieses Namens vorliegen.
grenade_launcher_x - X-Position des Granatwerfersymbols auf dem Waffensymbol im Inventar.
grenade_launcher_y - Y-"""

grenade_class - Liste der Munitionstypen, die der Granatwerfer verschießen kann.
grenade_flame_particles - Name des Partikeleffekts, der durch einen Schuss aktiviert wird.
snd_shoot_grenade - Name des Sounds, der durch einen Schuss aktiviert wird.
snd_reload_grenade - Name des Sounds, der durch Nachladen des Granatwerfers aktiviert wird.

Die Liste mit den Parametern ist von Krishty. Hab selber vor Monaten die Lust an Stalker verloren und seit dem staubt der Post unvollständig vor sich hin. Sorry dafür. Jetzt ist er aber endlich vollständig. Vielen Dank für deine Arbeit Krishty

---

Slot einer Waffe ändern

Ok, hier nochmal für alle, wie man die Slot-Belegung einer Waffe ändert.
[Beispiel anhand der MP5]

• gamedata\config\weapons\w_mp5.ltx
Nach "slot" suchen und gleich "1" setzen.

=> slot = 1 ; // secondary [bzw. is dann primary]

• gamedata\config\misc\unique_items.ltx
Hier einmal "[wpn_mp5_m1]:wpn_mp5" und einmal nach "[wpn_mp5_m2]:wpn_mp5" suchen und jeweils vor
"condition_shot_dec = 0.00003"
eine neue Zeile mit "slot = 1" einfügen.

Die normale MP5, sowie die MP5 mit 9x18 Munition ([wpn_mp5_m1]:wpn_mp5) benötigen je nur 3 Zellen im Inventar.

Die Silenced MP5 ([wpn_mp5_m2]:wpn_mp5), und eine mit Schalldämpfer nachgerüstete MP5 benötigen dagegen 4 Zellen.
[Hier gilt es auch bei anderen Waffen darauf zu achten, wie viel Platz sie benötigen. Allein, sowie mit Schalldämpfer]

Um das Problem zu umgehen, wird der Pistolenslot um eine Zelle erweitert.

• "gamedata\config\ui\inventory_new.xml" und dort nach dem Pistolen-Slot suchen
("dragdrop_pistol") und das Ganze mit dem hier ersetzen [man kann aber auch mit der Zellbreite spielen]:

Code:
<dragdrop_pistol x="38" y="118" width="156" height="104"
cell_width = "39" cell_height="39" rows_num="2" cols_num="4"
custom_placement="0"/>Dann hat man halt 4x2 Zellen im Pistolenslot. Auch nicht sooo schlimm, wie ich finde.
Viel Spaß damit.
DerSchwabe ist offline

So, hier mal eine Anleitung, wie man einen NPC zum Händler macht

Als Beispiel: Neuer Händler "Ratcatcher" im Agroprom

Zu ändernde Files:
- config\scripts\agr\agr_ratcatcher.ltx
- config\creatures\m_stalker.ltx

Neu anzulegende File:
- config\misc\trade_military.ltx

Wenn eigener Dialog dabei:
- config\gameplay\dialogs_agroprom.ltx
- config\gameplay\info_l03agroprom.xml [Wenn Dialog nur einmal zu lesen sein soll]
- config\text\ger\stable_dialogs_agroprom.xml

config\scripts\agr\agr_ratcatcher.ltx
Den [logic] – Block um folgende zwei Zeilen erweitern:

trade=misc\trade_military.ltx
inv_max_weight = 10000

config\creatures\m_stalker.ltx
use_single_item_rule = on => off [Nötig, dass NPC mehr als eine Waffe tragen kann (wirkt sich auf alle Stalker dann aus)]

config\misc\trade_military.ltx
Neue Trader-File, Angebot kann mit Hilfe der zu erfüllenden Aufträge anderer Trader gestaffelt werden.
Am einfachsten ist es, die trade_trader.ltx zu kopieren und umzubenennen und entsprechend das Angebot anzupassen.

Dialog
Bei Wunsch kann eigener Dialog geschrieben werden, wenn ein Marker gesetzt werden soll, sodass eine das / ein bestimmtes Gespräch nur einmal vorkommt, muss eine info_portion in der info_l03agrprom.xml angelegt werden und entsprechend abgefragt werden.

Bsp: Diesen Dialog-Logik-Block am Ende in die config\gameplay\dialogs_agroprom.ltx kopieren [vor </game_dialogs>]

Code:

```
<dialog id="ratcatcher_trader">
        <dont_has_info>agr_ratcatcher_start</dont_has_info>
      <phrase_list>
        <phrase id="0">
          <text>ratcatcher_trader_0</text>
          <next>1</next>
        </phrase>
        <phrase id="1">
                <text>ratcatcher_trader_1</text>
```

```
            <next>2</next>
         </phrase>
         <phrase id="2">
            <text>ratcatcher_trader_2</text>
            <next>3</next>
         </phrase>
         <phrase id="3">
            <text>ratcatcher_trader_3</text>
            <next>4</next>
            <next>41</next>
         </phrase>
         <phrase id="4">
            <text>ratcatcher_trader_4</text>
         </phrase>
         <phrase id="41">
                  <precondition>agroprom_dialog.actor_have_stuff</precondition>
            <text>ratcatcher_trader_41</text>
            <action>agroprom_dialog.transfer_stuff</action>
            <next>5</next>
         </phrase>
         <phrase id="5">
                  <give_info>agr_ratcatcher_start</give_info>
            <text>ratcatcher_trader_5</text>
         </phrase>
      </phrase_list>
   </dialog>
```
Das Gespräch findet einmal start und dann nicht mehr, gemacht wird das über die Abfrage nach der Info agr_ratcatcher_start und dem Setzen dieser Info.

Deklariert werden muss diese in der - config\gameplay\info_l03agroprom.xml
Hier fügt man am Ende einfach folgende Zeile mit ein:
<info_portion id="agr_ratcatcher_start"></info_portion>

In diesem Fall möchte der Händler noch etwas von mir, dafür wurden zwei Funktionen in das scripts\agroprom_dialog.script eingefügt:

Code:
```
function actor_have_stuff(first_speaker, second_speaker)
        return  first_speaker:object("bread") ~= nil and
                        first_speaker:object("conserva") ~= nil and
                        first_speaker:money() >= 1000
end

function transfer_stuff(first_speaker, second_speaker)
        dialogs.relocate_item_section(second_speaker, "bread", "out")
        dialogs.relocate_item_section(second_speaker, "conserva", "out")
        dialogs.relocate_money(second_speaker, 1000, "out")
end
```
Was jetzt noch Spielerei ist, ist das Angebot abhängig von dem neuen Dialog zu machen. Sprich ich bekomme erst Ware, wenn er bezahlt wurde. Wie und ob das geht, muss ich erst noch testen. ;-)

// EDIT:
Also, man kann das Angebot abhängig von der info_portion machen. Die trader_military.ltx sieht

dann zB so aus:

Code:
[trader]
buy_condition = trader_generic_buy
sell_condition = {+agr_ratcatcher_start} trader_test_sell, trader_start_sell
buy_supplies = {+agr_ratcatcher_start} supplies_test, supplies_start

Hier dann, womit gehandelt wird.

Und hier dann die Bedingungen was wann vorrätig und wie verkauft wird.
[supplies_start]

[supplies_test]
dolg_outfit = 20, 1
wpn_bm16 = 10,1
wpn_ak74u = 10,1

[trader_start_sell]

[trader_test_sell]
dolg_outfit = 1, 1
wpn_bm16 = 10,1
wpn_ak74u = 10,1Somit hat er am Anfang kein Angebot. Beendet man den zusätzlichen Dialog [und bezahlt den armen Kerl ;-)] erweitert er sein Angebot. Damit kann man jetzt schön spielen. Dialoge erst dann, wenn Quests erledigt sind [bzw die Labore], Angebotserweiterung ebenso erst danach, oder eben nach weiteren Bezahlungen.

In der config\text\ger\stable_dialogs_agroprom.xml muss jetzt nur noch ein Dialog angelegt werden.

Hier als Bsp:

Code:
<string id="ratcatcher_trader_0">
<text>Weshalb sollte ich Dich nicht erschießen?</text>
</string>
<string id="ratcatcher_trader_1">
<text>Weil...weil ich Dir helfen könnte. Ich habe immernoch begrenzten Zugang zu Militärausrüstung. Waffen zwar weniger, da die hier viel gebraucht werden, aber ich kann Dir Militär-Anzüge beschaffen.</text>
</string>
<string id="ratcatcher_trader_2">
<text>Und das würdest Du einfach so tun? Dich für mich in Gefahr begeben, um an das Zeug zu kommen, nur dass ich Dich nicht erschieße?</text>
</string>
<string id="ratcatcher_trader_3">
<text>Naja...ich könnte vielleicht auch ein wenig Geld und etwas zu Essen

gebrauchen. Sagen wir 1000 Rubel, eine Dose Wurst und ein Brot dazu. Diese andauernden Rattenburger bekommen mir nicht so gut - auch wenn in Filmen anderes behauptet wird.</text>
</string>
<string id="ratcatcher_trader_4">
<text>Das ist Wucher! Ich sollte Dich hier und jetzt erschießen!</text>
</string>
<string id="ratcatcher_trader_41">
<text>Gut, ich werde Dir helfen. Nimm Dir, was du brauchst.</text>
</string>
<string id="ratcatcher_trader_5">
<text>Na also, ich wusste doch, dass wir uns einig werden. Ich danke Dir. Ich bin mir sicher, dass Du ein ganz großer in der Zone wirst. Schau doch immer mal wieder bei mir rein, ich versuche meine Ware immer etwas zu erweitern. Vielleicht ist dann auch mal das ein oder andere Brauchbare für Dich dabei.</text>
</string>Den Block fügt man einfach ans Ende mit an [vor </string_table>]

Reparatur
Soll es dem NPC noch möglich sein, eine Reparatur durchzuführen, müssen zusätzlich 4 Files editiert werden:
- config\gameplay\character_desc_agroprom.xml
- config\gameplay\dialogs_repair
- config\text\ger\ stable_repair_dialogs.xml
- scripts\agroprom_dialog.script

config\gameplay\character_desc_agroprom.xml
Hier nach <actor_dialog>ratcatcher_trader</actor_dialog> suchen und in einer neuen Zeile folgendes einfügen:
<actor_dialog>ratcatcher_trader_repair</actor_dialog>

config\gameplay\dialogs_repair
Hier kopiert man sich jetzt einen Dialog [Bsp. anhand Sido] von
<dialog id="escape_trader_repair"> bis zum Ende des Dialogblocks [also nichts vom Barman noch mitkopieren] und fügt diesen Block am Ende noch einmal ein.
die id benennt man entsprechend in ratcatcher_trader_repair um. Danach ersetzt man alle escape_traderdurch ratcatcher_trader [natürlich nur im Dialogblock des Ratcatchers]
und alle escape_dialog. durch agroprom_dialog.

config\text\ger\ stable_repair_dialogs.xml
Hier kopiert man wieder den Teil von Sido und kopiert ihn ans Ende und ersetzt auch hier escape_trader durch ratcatcher_trader. Dann mal durchlesen und bei Bedarf den Text anpassen [zB. Ansprache mit Sidorovich]

scripts\agroprom_dialog.script
Hierfür muss man zusätzlich noch das escape_dialog.script im selbigen Ordner öffnen, und dort den ganzen Repair-Block am Anfang rauskopieren und in das agroprom_dialog.script einfügen.

Modifikation der Reparatur:

Händler können nur noch bestimmte Dinge reparieren

AKTUALISIERUNG 25.2.08
Man hat jetzt einen finanziellen Vorteil, wenn man einem Händler [hier eigentlich nur Freiheit und Wächter releveant] freundlich gesinnt ist.

Ich habe das Reparatur-Script umgeschrieben, sodass man einem Händler sagen kann, was er reparieren soll, und was nicht. Bsp. Anhand des escape_dialog.script
Natürlich kann man das dann je Händler beliebig anpassen.
Die Reparatur-Scripts der Standardhändler findet man hier:
scripts\...
escape_dialog.script
bar_dialog.script
dialogs_military.script
dialogs_yantar.script

Code:

```
-------------------------------------------------------------------------------------------------
-- Trader Repair
-------------------------------------------------------------------------------------------------

local weapon_profit_margin
local armor_profit_margin

-- Wenn ALLES REPARIERT werden soll, muss im Table "_" eingetragen werden !!
local repairTable = {
"wpn_walther", "wpn_pb", "wpn_pm", "wpn_mp5", "wpn_fort",
"wpn_bm16",
"novice"
}

local j = table.getn(repairTable)
local var_1 = ""
local var_2 = ""
local var_6 = ""

function trader_repair_precond(trader, actor)
	local item_in_slot_1 = db.actor:item_in_slot(1)
	local item_in_slot_2 = db.actor:item_in_slot(2)
	local item_in_slot_6 = db.actor:item_in_slot(6)


	if actor:relation(trader) == game_object.friend	then
		weapon_profit_margin = 10
		armor_profit_margin = 10
	else
		weapon_profit_margin = 13.5
		armor_profit_margin = 13.5
```

```
	end

	for i=1,j do
		if	item_in_slot_1 ~= nil and 99999 > (item_in_slot_1:condition() * 100000) and
			string.find(item_in_slot_1:name(), repairTable[i]) ~= nil then
				return true
			elseif	item_in_slot_2 ~= nil and 99999 > (item_in_slot_2:condition() *
100000) and
					string.find(item_in_slot_2:name(), repairTable[i]) ~= nil then
				return true
			elseif	item_in_slot_6 ~= nil and 99999 > (item_in_slot_6:condition() *
100000) and
					string.find(item_in_slot_6:name(), repairTable[i]) ~= nil then
				return true
		end
	end
	return false

end

function trader_check_money_s1(trader, actor)
	local item_in_slot_1 = db.actor:item_in_slot(1)

	for i=1,j do
		if item_in_slot_1 ~= nil and string.find(item_in_slot_1:name(), repairTable[i]) ~= nil
then
			local item_repair_cost = math.floor( (1-item_in_slot_1:condition()) *
item_in_slot_1:cost() * weapon_profit_margin )
			if item_repair_cost > 0 and db.actor:money() >= item_repair_cost and 1 >
item_in_slot_1:condition() then
				return true
			end
		end
	end
	return false
end

function trader_check_money_s2(trader, actor)
	local item_in_slot_2 = db.actor:item_in_slot(2)

	for i=1,j do
		if item_in_slot_2 ~= nil and string.find(item_in_slot_2:name(), repairTable[i]) ~= nil
then
			local item_repair_cost = math.floor( (1-item_in_slot_2:condition()) *
item_in_slot_2:cost() * weapon_profit_margin )
			if item_repair_cost > 0 and db.actor:money() >= item_repair_cost and 1 >
item_in_slot_2:condition() then
				return true
			end
		end
	end
	return false
```

```
end

function trader_check_money_s6(trader, actor)
	local item_in_slot_6 = db.actor:item_in_slot(6)

	for i=1,j do
		if item_in_slot_6 ~= nil and string.find(item_in_slot_6:name(), repairTable[i]) ~= nil
then
			local item_repair_cost = math.floor( (1-item_in_slot_6:condition()) *
item_in_slot_6:cost() * armor_profit_margin )
			if item_repair_cost > 0 and db.actor:money() >= item_repair_cost and 1 >
item_in_slot_6:condition() then
				return true
			end
		end
	end
	return false
end

function repair_costs(first_speaker, second_speaker)
	local task_texture, task_rect = get_texture_info("ui_iconsTotal_lost_money")
	local item_name_and_price = ""
	local item_repair_cost = 0


	if db.actor ~= nil then
		local item_in_slot_1 = db.actor:item_in_slot(1)
		local item_in_slot_2 = db.actor:item_in_slot(2)
		local item_in_slot_6 = db.actor:item_in_slot(6)

		for i=1,j do
			if item_in_slot_1 ~= nil and string.find(item_in_slot_1:name(),
repairTable[i]) ~= nil then
				var_1 = true
			end
			if item_in_slot_2 ~= nil and string.find(item_in_slot_2:name(),
repairTable[i]) ~= nil then
				var_2 = true
			end
			if item_in_slot_6 ~= nil and string.find(item_in_slot_6:name(),
repairTable[i]) ~= nil then
				var_6 = true
			end
		end


		if var_1 == true then
				item_repair_cost = math.floor( (1-item_in_slot_1:condition()) *
item_in_slot_1:cost() * weapon_profit_margin )
			if item_repair_cost > 0 then
				item_name_and_price =
game.translate_string("list_trader_repair_0").." %c[255,238,155,23]"..item_repair_cost.." Ru
```

```
%c[default]"
                                db.actor:give_talk_message(item_name_and_price, task_texture,
task_rect, "iconed_trade_info")
                        end
                end
                if var_2 == true then
                        item_repair_cost = math.floor( (1-item_in_slot_2:condition()) *
item_in_slot_2:cost() * weapon_profit_margin )
                        if item_repair_cost > 0 then
                                item_name_and_price =
game.translate_string("list_trader_repair_1").." %c[255,238,155,23]"..item_repair_cost.." Ru
%c[default]"
                                db.actor:give_talk_message(item_name_and_price, task_texture,
task_rect, "iconed_trade_info")
                        end
                end
                if var_6 == true then
                                item_repair_cost = math.floor( (1-item_in_slot_6:condition()) *
item_in_slot_6:cost() * armor_profit_margin )
                        if item_repair_cost > 0 then
                                item_name_and_price =
game.translate_string("list_trader_repair_2").." %c[255,238,155,23]"..item_repair_cost.." Ru
%c[default]"
                                db.actor:give_talk_message(item_name_and_price, task_texture,
task_rect, "iconed_trade_info")
                        end
                end
        end
end



function trader_repiar_weapon_s1(trader, actor)
        local item_in_slot = db.actor:item_in_slot(1)
        if item_in_slot ~= nil then
                local item_repair_cost = math.floor( (1-item_in_slot:condition()) *
item_in_slot:cost() * weapon_profit_margin )
                item_in_slot:set_condition(1)
                dialogs.relocate_money(actor, item_repair_cost, "out")
        end
end

function trader_repiar_weapon_s2(trader, actor)
        local item_in_slot = db.actor:item_in_slot(2)
        if item_in_slot ~= nil then
                local item_repair_cost = math.floor( (1-item_in_slot:condition()) *
item_in_slot:cost() * weapon_profit_margin )
                item_in_slot:set_condition(1)
                dialogs.relocate_money(actor, item_repair_cost, "out")
        end
end
```

```
function trader_repiar_armor_s6(trader, actor)
	local item_in_slot = db.actor:item_in_slot(6)
	if item_in_slot ~= nil then
		local item_repair_cost = math.floor( (1-item_in_slot:condition()) *
item_in_slot:cost() * armor_profit_margin )
		item_in_slot:set_condition(1)
		dialogs.relocate_money(actor, item_repair_cost, "out")
	end
end
```

Es wird ein repairTable angelegt, in dem alles festgelegt wird, was repariert werden darf. [wenn alles, muss "_" eingetragen werden].

Hab das jetzt aufgeteilt in 3 Zeilen: Slot 1, Slot 2 und Anzug - dient lediglich der Übersicht. Es muss nicht die jeweilige Bezeichnung der Waffe stimmen, es langt, wenn der angegebene Ausdruck im Waffennamen gefunden wird!
Es wird also die Standard-Fort und die Modifizierte repariert! Genauso beim Anzug, es werden alle Anzüge repariert, die "Novice" im Namen tragen.

Ich habe Geizhals[Händler der Freiheitler] und Screw ["Schrauber" - Freiheitler] so eingerichtet, dass sie Reparaturen durchführen können. Da ich diesen einen eigenen repairTable, sowie eigene Reparaturpreise gegeben habe, musste das Script ein wenig erweitert werden, sodass sie nicht auf ein und die selbe function innerhalb des Scripts zugreifen müssen.
Hier als Bsp. das military_dialogs.script

Code:

```
---------------------------------------------------------------------------------------------------
-- Trader Repair
---------------------------------------------------------------------------------------------------

local weapon_profit_margin
local armor_profit_margin

-- Wenn ALLES REPARIERT werden soll, muss im Table "_" eingetragen werden !!
local repairTable = {
"wpn_walther", "wpn_pb", "wpn_pm", "wpn_fort", "wpn_hpsa", "wpn_colt", "wpn_beretta",
"wpn_mp5", "eagle",
"wpn_sig",
"wpn_svu", "wpn_svd", "wpn_fn", "wpn_g36", "wpn_lr300", "wpn_l85", "wpn_rg", "wpn_rpg",
"svoboda", "freedom_scientific_outfit"
                                        }

local repairTable_screw = {
"wpn_ak", "wpn_abakan",
"specnaz", "soldier", "military", "specops", "neytral_exo_antigas_outfit"
                                        }

local j = table.getn(repairTable)
local k = table.getn(repairTable_screw)
local var_1 = ""
```

```
local var_2 = ""
local var_6 = ""


-- GEIZHALS --
function trader_repair_precond(trader, actor)
	local item_in_slot_1 = db.actor:item_in_slot(1)
	local item_in_slot_2 = db.actor:item_in_slot(2)
	local item_in_slot_6 = db.actor:item_in_slot(6)

	if actor:relation(trader) == game_object.friend	then
		weapon_profit_margin = 10
		armor_profit_margin = 10
	else
		weapon_profit_margin = 13.5
		armor_profit_margin = 13.5
	end


	for i=1,j do
		if	item_in_slot_1 ~= nil and 99999 > (item_in_slot_1:condition() * 100000) and
			string.find(item_in_slot_1:name(), repairTable[i]) ~= nil then
				return true
			elseif	item_in_slot_2 ~= nil and 99999 > (item_in_slot_2:condition() *
100000) and
					string.find(item_in_slot_2:name(), repairTable[i]) ~= nil then
				return true
			elseif	item_in_slot_6 ~= nil and 99999 > (item_in_slot_6:condition() *
100000) and
					string.find(item_in_slot_6:name(), repairTable[i]) ~= nil then
				return true
		end
	end
	return false

end

function trader_check_money_s1(trader, actor)
	local item_in_slot_1 = db.actor:item_in_slot(1)

	for i=1,j do
		if item_in_slot_1 ~= nil and string.find(item_in_slot_1:name(), repairTable[i]) ~= nil
then
			local item_repair_cost = math.floor( (1-item_in_slot_1:condition()) *
item_in_slot_1:cost() * weapon_profit_margin )
			if item_repair_cost > 0 and db.actor:money() >= item_repair_cost and 1 >
item_in_slot_1:condition() then
				return true
			end
		end
	end
	return false
```

```
end

function trader_check_money_s2(trader, actor)
	local item_in_slot_2 = db.actor:item_in_slot(2)

	for i=1,j do
		if item_in_slot_2 ~= nil and string.find(item_in_slot_2:name(), repairTable[i]) ~= nil
then
			local item_repair_cost = math.floor( (1-item_in_slot_2:condition()) *
item_in_slot_2:cost() * weapon_profit_margin )
			if item_repair_cost > 0 and db.actor:money() >= item_repair_cost and 1 >
item_in_slot_2:condition() then
				return true
			end
		end
	end
	return false
end

function trader_check_money_s6(trader, actor)
	local item_in_slot_6 = db.actor:item_in_slot(6)

	for i=1,j do
		if item_in_slot_6 ~= nil and string.find(item_in_slot_6:name(), repairTable[i]) ~= nil
then
			local item_repair_cost = math.floor( (1-item_in_slot_6:condition()) *
item_in_slot_6:cost() * armor_profit_margin )
			if item_repair_cost > 0 and db.actor:money() >= item_repair_cost and 1 >
item_in_slot_6:condition() then
				return true
			end
		end
	end
	return false
end

function repair_costs(first_speaker, second_speaker)
	local task_texture, task_rect = get_texture_info("ui_iconsTotal_lost_money")
	local item_name_and_price = ""
	local item_repair_cost = 0


	if db.actor ~= nil then
		local item_in_slot_1 = db.actor:item_in_slot(1)
		local item_in_slot_2 = db.actor:item_in_slot(2)
		local item_in_slot_6 = db.actor:item_in_slot(6)

		for i=1,j do
			if item_in_slot_1 ~= nil and string.find(item_in_slot_1:name(),
repairTable[i]) ~= nil then
				var_1 = true
			end
```

```
                        if item_in_slot_2 ~= nil and string.find(item_in_slot_2:name(),
repairTable[i]) ~= nil then
                                var_2 = true
                        end
                        if item_in_slot_6 ~= nil and string.find(item_in_slot_6:name(),
repairTable[i]) ~= nil then
                                var_6 = true
                        end
                end


                if var_1 == true then
                                item_repair_cost = math.floor( (1-item_in_slot_1:condition()) *
item_in_slot_1:cost() * weapon_profit_margin )
                        if item_repair_cost > 0 then
                                item_name_and_price =
game.translate_string("list_trader_repair_0").." %c[255,238,155,23]"..item_repair_cost.." Ru
%c[default]"
                                db.actor:give_talk_message(item_name_and_price, task_texture,
task_rect, "iconed_trade_info")
                        end
                end
                if var_2 == true then
                        item_repair_cost = math.floor( (1-item_in_slot_2:condition()) *
item_in_slot_2:cost() * weapon_profit_margin )
                        if item_repair_cost > 0 then
                                item_name_and_price =
game.translate_string("list_trader_repair_1").." %c[255,238,155,23]"..item_repair_cost.." Ru
%c[default]"
                                db.actor:give_talk_message(item_name_and_price, task_texture,
task_rect, "iconed_trade_info")
                        end
                end
                if var_6 == true then
                                item_repair_cost = math.floor( (1-item_in_slot_6:condition()) *
item_in_slot_6:cost() * armor_profit_margin )
                        if item_repair_cost > 0 then
                                item_name_and_price =
game.translate_string("list_trader_repair_2").." %c[255,238,155,23]"..item_repair_cost.." Ru
%c[default]"
                                db.actor:give_talk_message(item_name_and_price, task_texture,
task_rect, "iconed_trade_info")
                        end
                end
        end
end

-- SCREW --
function trader_repair_precond_screw(trader, actor)
        local item_in_slot_1 = db.actor:item_in_slot(1)
        local item_in_slot_2 = db.actor:item_in_slot(2)
        local item_in_slot_6 = db.actor:item_in_slot(6)
```

```
	if actor:relation(trader) == game_object.friend	then
		weapon_profit_margin = 12
		armor_profit_margin = 12
	else
		weapon_profit_margin = 16
		armor_profit_margin = 16
	end

	for i=1,k do
		if	item_in_slot_1 ~= nil and 99999 > (item_in_slot_1:condition() * 100000) and
			string.find(item_in_slot_1:name(), repairTable_screw[i]) ~= nil then
				return true
			elseif	item_in_slot_2 ~= nil and 99999 > (item_in_slot_2:condition() *
100000) and
					string.find(item_in_slot_2:name(), repairTable_screw[i]) ~=
nil then
				return true
			elseif	item_in_slot_6 ~= nil and 99999 > (item_in_slot_6:condition() *
100000) and
					string.find(item_in_slot_6:name(), repairTable_screw[i]) ~=
nil then
				return true
		end
	end
	return false

end

function trader_check_money_s1_screw(trader, actor)
	local item_in_slot_1 = db.actor:item_in_slot(1)

	for i=1,k do
		if item_in_slot_1 ~= nil and string.find(item_in_slot_1:name(), repairTable_screw[i])
~= nil then
			local item_repair_cost = math.floor( (1-item_in_slot_1:condition()) *
item_in_slot_1:cost() * weapon_profit_margin )
			if item_repair_cost > 0 and db.actor:money() >= item_repair_cost and 1 >
item_in_slot_1:condition() then
				return true
			end
		end
	end
	return false
end

function trader_check_money_s2_screw(trader, actor)
	local item_in_slot_2 = db.actor:item_in_slot(2)

	for i=1,k do
		if item_in_slot_2 ~= nil and string.find(item_in_slot_2:name(), repairTable_screw[i])
~= nil then
```

```
                        local item_repair_cost = math.floor( (1-item_in_slot_2:condition()) *
item_in_slot_2:cost() * weapon_profit_margin )
                        if item_repair_cost > 0 and db.actor:money() >= item_repair_cost and 1 >
item_in_slot_2:condition() then
                                return true
                        end
                end
        end
        return false
end

function trader_check_money_s6_screw(trader, actor)
        local item_in_slot_6 = db.actor:item_in_slot(6)

        for i=1,k do
                if item_in_slot_6 ~= nil and string.find(item_in_slot_6:name(), repairTable_screw[i])
~= nil then
                        local item_repair_cost = math.floor( (1-item_in_slot_6:condition()) *
item_in_slot_6:cost() * armor_profit_margin )
                        if item_repair_cost > 0 and db.actor:money() >= item_repair_cost and 1 >
item_in_slot_6:condition() then
                                return true
                        end
                end
        end
        return false
end

function repair_costs_screw(first_speaker, second_speaker)
        local task_texture, task_rect = get_texture_info("ui_iconsTotal_lost_money")
        local item_name_and_price = ""
        local item_repair_cost = 0
        local var_1 = ""
        local var_2 = ""
        local var_6 = ""


        if db.actor ~= nil then
                local item_in_slot_1 = db.actor:item_in_slot(1)
                local item_in_slot_2 = db.actor:item_in_slot(2)
                local item_in_slot_6 = db.actor:item_in_slot(6)

                for i=1,k do
                        if item_in_slot_1 ~= nil and string.find(item_in_slot_1:name(),
repairTable_screw[i]) ~= nil then
                                var_1 = true
                        end
                        if item_in_slot_2 ~= nil and string.find(item_in_slot_2:name(),
repairTable_screw[i]) ~= nil then
                                var_2 = true
                        end
                        if item_in_slot_6 ~= nil and string.find(item_in_slot_6:name(),
```

```
repairTable_screw[i]) ~= nil then
                                var_6 = true
                        end
                end


                if var_1 == true then
                                item_repair_cost = math.floor( (1-item_in_slot_1:condition()) *
item_in_slot_1:cost() * weapon_profit_margin )
                        if item_repair_cost > 0 then
                                item_name_and_price =
game.translate_string("list_trader_repair_0").." %c[255,238,155,23]"..item_repair_cost.." Ru
%c[default]"
                                db.actor:give_talk_message(item_name_and_price, task_texture,
task_rect, "iconed_trade_info")
                        end
                end
                if var_2 == true then
                        item_repair_cost = math.floor( (1-item_in_slot_2:condition()) *
item_in_slot_2:cost() * weapon_profit_margin )
                        if item_repair_cost > 0 then
                                item_name_and_price =
game.translate_string("list_trader_repair_1").." %c[255,238,155,23]"..item_repair_cost.." Ru
%c[default]"
                                db.actor:give_talk_message(item_name_and_price, task_texture,
task_rect, "iconed_trade_info")
                        end
                end
                if var_6 == true then
                                item_repair_cost = math.floor( (1-item_in_slot_6:condition()) *
item_in_slot_6:cost() * armor_profit_margin )
                        if item_repair_cost > 0 then
                                item_name_and_price =
game.translate_string("list_trader_repair_2").." %c[255,238,155,23]"..item_repair_cost.." Ru
%c[default]"
                                db.actor:give_talk_message(item_name_and_price, task_texture,
task_rect, "iconed_trade_info")
                        end
                end
        end
end




function trader_repiar_weapon_s1(trader, actor)
        local item_in_slot = db.actor:item_in_slot(1)
        if item_in_slot ~= nil then
                local item_repair_cost = math.floor( (1-item_in_slot:condition()) *
item_in_slot:cost() * weapon_profit_margin )
                item_in_slot:set_condition(1)
                dialogs.relocate_money(actor, item_repair_cost, "out")
```

```
	end
end

function trader_repiar_weapon_s2(trader, actor)
	local item_in_slot = db.actor:item_in_slot(2)
	if item_in_slot ~= nil then
		local item_repair_cost = math.floor( (1-item_in_slot:condition()) *
item_in_slot:cost() * weapon_profit_margin )
		item_in_slot:set_condition(1)
		dialogs.relocate_money(actor, item_repair_cost, "out")
	end
end

function trader_repiar_armor_s6(trader, actor)
	local item_in_slot = db.actor:item_in_slot(6)
	if item_in_slot ~= nil then
		local item_repair_cost = math.floor( (1-item_in_slot:condition()) *
item_in_slot:cost() * armor_profit_margin )
		item_in_slot:set_condition(1)
		dialogs.relocate_money(actor, item_repair_cost, "out")
	end
endBis auf die 3 eigentlichen Reparatur-functions wurden alle kopiert und an den entsprechenden NPC angepasst.
Auch wichtig: In der dialogs_repair.xml müssen dann die Funktionsnamen beim jeweiligen NPC angepasst werden! Geizhals geht auf die ursprüngliche function function trader_repair_precond und Screw auf function trader_repair_precond_screw. So lassen sich alle Sachen wunderbar trennen.
Im Bar-Level ist es das selbe Spiel, dort reparieren bei mir Petrenko [Händler der Wächter] und der Wirt.
```

Hier noch eine Übersicht der Waffen und Anzüge [Anzüge nach Fraktion geordnet] - Angaben der Waffen und Anzüge wurden aus der INVASION 1.1 entnommen !!

Reichweite der KI erhöhen
_____von Krishty

Standardmäßig werden Gegner in Stalker erst sichtbar, wenn sie weniger als 100m entfernt sind. Wenn man sich dann wieder weiter als 150m von ihnen entfernt verschwinden sie wieder. Hier schildere ich eine Möglichkeit, auch über größere Distanzen mit Monstern und NPCs zu interagieren.

Vorbedingungen:
Selbstverständlich entpackte gamedata (siehe Allgemeines im Eingangspost dieses Threads).
Ich habe die Änderungen an der Spielversion 1.0006 getestet, sie sollten aber auch mit früheren Versionen des Spiels funktionieren. Der neueste Patch wird also nicht zwingend benötigt.

Einleitung:
Alle NPCs und Monster in Stalker werden vom ALife-System gesteuert. Ein besonderes Merkmal dieses Systems ist das Level-of-Detail: Nur die Lebewesen in näherer Umgebung des Spielers werden mit "voller Intelligenz" gesteuert.
Jene, die zu weit weg sind als dass der Spieler sie sehen könnte, werden mit simpleren Algorithmen gesteuert. Beispielsweise muss für einen Hund, der drei Level vom Spieler entfernt durch einen Wald streift, nicht berechnet werden wo er welchen Fuß hinsetzt, genauso wenig wird er Bild für Bild mitgezeichnet. Stattdessen werden nur seine ungefähren Bewegungen berechnet und erst wenn sich der Spieler nähert wird auf "volle" KI umgeschaltet - und der Hund für den Spieler sichtbar.
So ist es möglich, dass sich die hunderten Lebewesen überall in der Zone ständig bewegen und ihren gewohnten Tätigkeiten nachgehen, sich der Rechenaufwand aber dennoch in einem Umfang hält, der von Mittelklasse-Prozessoren bewältigt werden kann - nebenbei, während sich der Spieler ein paar Level weiter durch Mutantenhorden kämpft.

ALife:
Um über größere Distanzen mit NPCs interagieren zu können sollte unser erster Schritt sein, die Distanz, ab welcher Lebewesen mit voller KI berechnet und sichtbar werden, zu erhöhen. Den entsprechenden Eintrag finden wir in gamedata\config\alife.ltx: Fast ganz unten, in der Section alife, finden wir den Eintrag switch_distance. Er kontrolliert, ab wieviel Metern Entfernung des Lebewesens zum Spieler das Alife-System auf volle Leistung umschaltet.
Natürlich bedeuten größere Werte auch, dass der Prozessor stärker belastet wird. Aber schon ab einem Core2Duo sollten Werte bis 500m kein Problem darstellen. Für dieses Tutorial setze ich 400m ein:
Code:
```
[alife]
...
        switch_distance = 400 ; meters
```
...Wenn wir Stalker jetzt starten, werden uns zwei schwerwiegende Probleme auffallen: Dass wir gar keine 400 Meter weit sehen können und die NPCs auch nicht. Darum kümmern wir uns in den nächsten beiden Abschnitten.

Sichtweite:
Die Sichtweite ist in Stalker von Werk aus sehr gering. Nun werden NPCs zwar schon auf 400m dargestellt, verschwinden aber komplett im Nebel... was tun?
Die Sichtweite lässt sich in den Wetterbeschreibungen einstellen. Wir finden sie in gamedata\config\weathers\weather_default.ltx. Entscheidend sind für jede Wetterlage zwei Werte:
far_plane gibt an, bis zu welcher Entfernung die Grafikengine zeichnet, in Metern. Ab dem ersten Meter hinter diesem Wert wird alles abgeschnitten und durch die Skybox ersetzt.
fog_distance gibt an, bis zu welcher Entfernung, in Metern, der Nebel immer dichter wird. Ab der angegebenen Entfernung verschwindet alles komplett im Nebel.
Wir können bei beiden das gleiche eintragen: 600m reichen um das komplette Level zu überblicken, uns reichen an dieser Stelle 400m, die Entfernung ab der auch Lebewesen sichtbar werden.
Beachte: Diese Werte kommen 24 Mal vor, für jede Stunde des Tages, und müssen auch jedes Mal geändert werden.
Wenn wir das Spiel nun starten sollten wir eine hervorragende Weitsicht haben. Die Stalker allerdings werden sich immernoch verhalten, als sei ihnen alles was weiter als 100m weg ist, egal.

KI-Sichtweite:

Nun, da wir andere Stalker auf 400m erkennen können, wäre es nur fair dass auch NPCs ihre Freunde, ihre Feinde und uns selbst auf 400m erkennen können. Ganz klar: Wir müssen der KI eine Brille spendieren. Dazu bewegen wir uns in die gamedata\config\creatures\m_stalker.ltx, die für alle menschlichen Lebewesen der Zone gilt. Lass dich nicht davon täuschen dass es für jede Fraktion eine ltx zu geben scheint, diese Dateien sind Überbleibsel einer alten Spielversion und werden ignoriert.
Achtung: Damit diese Änderungen übernommen werden, muss man ein neues Spiel beginnen!

In der Datei finden wir zwei Werte: eye_fov und eye_range. Ersterer gibt die Breite des Blickfeldes an, in Grad. 150 haben sich nach meiner Erfahrung als recht guter Wert ergeben. Letzterer gibt die Sichtweite an, in Metern. Hier tragen wir fairerweise 400 ein - die NPCs können nun genauso gut sehen wie wir. Oder?

Nein, wenn wir das Spiel jetzt starten beweisen die NPCs eine erstaunliche Sehstärke. Da braucht nur ein Soldat weit weg am Horizont aufzutauchen und alle drehen durch und fangen wie wild an zu ballern. Alle NPCs spielen total verrückt. Da muss noch was geändert werden!

Bevor wir loslegen, ein kurzer Ausflug in die Psyche der Stalker:
Im Grunde kennen alle NPCs zwei Gemütszustände - entspannt und angespannt.
Entspannt sind sie, wenn sie am Lagerfeuer sitzen und sich unterhalten, oder wenn sie Wache stehen und kein Feind weit und breit zu sehen ist.
Angespannt sind sie, sobald eine Kugel neben ihnen einschlägt, sobald sie selbst oder jemand in der Nähe verletzt wird oder sobald ein Feind auftaucht. Dieser angespannte Zustand schlägt erst wieder in Entspannung um, sobald dieser Feind getötet oder lange Zeit nichts mehr passiert ist.
Wir können nun für diese beiden Gemütszustände seperat festlegen, wie gut die Stalker jeweils sehen können.

Die entsprechenden Sections heißen stalker_vision_free für Entspannung und stalker_vision_danger für Anspannung. Jede von ihnen enthält u.a. min_view_distance, max_view_distance und always_visible_distance. Die ersten beiden geben an, welche Distanz der Stalker in seiner Gemütslage überblicken kann und always_visible_distance, ab welcher Entfernung er einen Feind bemerkt - egal wie gut getarnt er auch ist.
Das besondere an diesen Werten ist, dass sie relativ zu eye_range sind. Wir können dort also nicht einfach 200 für 200m eintragen. Wenn wir 200m meinen, müssen wir 0.5 eintragen, weil 0.5 * eye_range (400m) = 200m sind. Ach, hier einfach meine Werte:
Code:

```
[stalker_vision_free]
min_view_distance           = 0.05  ; 0.05 * 400m =  20m
max_view_distance           = 0.25  ; 0.25 * 400m = 100m
always_visible_distance             = 0.05  ; 0.05 * 400m =  20m
...

[stalker_vision_danger]
min_view_distance           = 0.10  ; 0.10 * 400m =  40m
max_view_distance           = 1.00  ; 1.00 * 400m = 400m
always_visible_distance             = 0.10  ; 0.10 * 400m =  40m
```

...Sehr gut kann man die Wirkung dieser Werte am Armeeposten im Kordon nachvollziehen. Starten wir also das Spiel.

Ergebnis:
Stellst du dich vor dem Dorf auf die Straße, unternehmen die Soldaten am Posten nichts, weil du

gut 150m weit weg bist - das ist mehr als max_view_distance im entspannten Zustand. Näherst du dich auf weniger als 100m, beginnen sie langsam, dich zu bemerken.
Neuer Versuch: Wieder stehst du 150m weit vom Armeeposten weg, wieder bemerken sie dich nicht. Du ziehst deine Waffe und feuerst ein paar Schuss in ihre Richtung - nun merken sie, dass sie in Gefahr sind. Sie sehen dich weil sie jetzt genau hinschauen - nämlich 400m weit, wie max_view_distance im angespannten Zustand - eröffnen das Feuer und verfolgen dich, bis du dich irgendwo dauerhaft verstecken kannst oder weiter als 400m weggelaufen bist.

Die 400m die ich in diesem Tutorial benutzt habe, sind schon recht viel und können manchmal zu Fehlern führen. Insbesondere nachdem man ein gespeichertes Spiel lädt sind NPCs oft im alarmierten Zustand, auch wenn sie während des Speicherns noch völlig ruhig waren. Falls sowas missionskritisch werden sollte, muss man die Sichtweite noch weiter reduzieren. Für Level, die nicht so weiträumig sind wie es bspw. das Armeelager ist, reichen auch 200m.

Bearbeitung der all.spawn mittels ACDC
von Gastredner

Die Bearbeitung der all.spawn ist nicht gerade ein Zuckerschlecken. Bereits die Bereistellung der Tools - also in erster Linie ACDC - bereitet so manchem Probleme.
Bisher gab es eine relativ gut verständliche - jedoch leider auch unvollständige - Anleitung, wie man ACDC ans Laufen bringt, auf dem Wiki zum S.T.A.L.K.E.R.-SDK(sdk.stalker-game.com), doch leider scheint der Inhalt der Seite derzeit verloren gegangen zu sein. Daher muss ich mich leider auch auf die Ausführungen zum grundlegenden Spawnen von Gegenständen beschränken, da ich selbst noch keine Zeit hatte, das Spawnen von NPCs, dem Beispiel des Wikis folgend, auszuprobieren und durchzutesten. Allerdings kann ich über den Cache noch auf die Seiten zugreifen und werde versuchen, möglichst große Teile davon zu sichern.

Fangen wir aber erst einmal mit ACDC an. Zunächst benötigt ihr zwei Dinge, um das Programm ans Laufen zu bringen: Einerseits eine Perl-Umgebung wie ActivePerl, andererseits natürlich das Programm selbst.
Dabei gibt es zu beachten, dass es ACDC für verschiedene Versionen von S.T.A.L.K.E.R. gibt. Am wichtigsten ist jedoch ohne Zweifel die aktuellste Version für v1.0004 von S.T.A.L.K.E.R.
Beziehen könnt ihr ACDC für diese Version von HIER.
Denkt daran: Diese Version funktioniert nur zusammen mit der v1.004 und v1.0005 von "S.T.A.L.K.E.R. - Shadow of Chernobyl." Auch mit Mods, die neue Gegenstände ins Spiel einführen bzw. spawnen ist diese Version von ACDC nicht kompatibel.

Fangen wir nun an: Als erstes installieren wir ActivePerl(es sei denn, ihr verfügt bereits über eine Perl-Umgebung auf eurem System). Entpackt dazu das Archiv und startet die installer.bat. Die Nachfragen solltet ihr alle mit "Ja" bestätigen, die Wahl des Installationsverzeichnisses ist euch überlassen.
Nachdem der Installer seine Arbeit getan hat, überprüfen wir, ob die Umgebungsvaribalen korrekt gesetzt sind - dies verursachte bei einigen Leute Probleme, durch die die Perl-Umgeung nicht korrekt lief.

Dazu gehen wir wie folgt vor: Startet eine Instanz der Eingabeaufforderung - am schnellsten geht das über den Ausführen-Dialog mit dem Befehl "cmd." In der sich öffnenden Shell gebt ihr "perl"

ein und drückt die Eingabetaste. Wenn nun gar nichts geschieht und einfach nur ein blinkender Cursor erscheint, ist Perl korrekt eingerichtet. Sollte hingegen die Fehlermeldung erscheinen, dass der Vefehl "perl" nicht gefunden werden könne, müsst ihr den Dialog zur Bearbeitung der Umgebungsvariablen aufrufen: Öffnet die Eigenschaften eures Arbeitsplatzes, geht zu dem Reiter "Erweitert" und klickt dort auf den Button "Umgebungsvariablen." Euch zeigt sich folgender Dialog:

Schaut euch die Liste der Systemvariablen an. Wenn ihr beim Installer von ActivePerl alle Nachfragen mit "Ja" beantwortet habt, solltet ihr dort bereits eine Variable namens "Path" finden, die - unter anderem - auch zwei Einträge in das Installationsverzeichnis von Perl beinhaltet. Ich selbst habe ActivePerl in das Verzeichnis C:\Perl installiert, daher lauten diese Umgebungsvariablen bei mir:

Code:
C:\Perl\site\bin;
C:\Perl\bin;Kopiert nun die beiden Variablen, die zu Perl führen, aus der "Path"-Variable des Systems in eure "PATH"-Benutzervariable. Achtet dabei darauf, keine anderen Einträge zu löschen - das könnte sich recht dramatisch auswirken.
Bestätigt die Änderungen und Perl sollte nun korrekt laufen. Überprüfen könnt ihr dies erneut über die Kommandokonsole.

Jetzt, wo ActivePerl läuft, wenden wir uns ACDC selbst zu. Extrahiert den Inhalt des Archives, am besten in einen eigenen Ordner. Ihr solltet nun folgendes sehen:

Als erstes benennen wir die Datei "acdc11oct.pl" um: Zu "acdc.pl"
Nun müssen wir zwei Batch-Dateien anlegen. Dazu öffnet ihr einen beliebigen Texteditor(WICHTIG: Kein Textverarbeitungsprogramm wie MS Word oder OOo Writer!) und kopiert folgenden Code in ihn:

Code:
perl acdc.pl -d all.spawnSpeichert die Datei nun als "decompile.bat" und wiederholt den Vorgang für die "compile.bat", diesmal mit folgendem Code:

Code:
@echo Compiling new.spawn...
@perl acdc.pl -c all.ltx -o new.spawn
@echo DoneDiese Dateien benötigen wir gleich, um die all.spawn in ihre Einzelteile zu zerlegen - zu "dekompilieren" - und um diese Einzelteile, nach erfolgreicher Manipulation, wieder zu einer einzigen all.spawn zusammenzuschmelzen.
Wer damit Probleme hat, kann sich die beiden Batch-Dateien auch HIER als selbstextrahierendes Archiv herunterladen.
Nun benötigen wir noch eine letzte "Zutat", bevor wir mit der Arbeit beginnen können: Die all.spawn. Extrahiert diese aus den .dba- und .dbb-Archiven in eurem S.T.A.L.K.E.R.-Hauptverzeichnis und kopiert sie in den acdc-Ordner. Dieser sollte nun folgendermaßen aussehen:

In Ordnung, fangen wir an: Startet die decompile.bat und wartet, bis der Vorgang abgeschlossen ist. Ihr findet nun eine Menge neuer Dateien vor: Die all.spawn. Ihr Inhalt: Eine alife.ltx und eine way.ltx für jedes Level im Spiel, außerdem eine Datei namens all.ltx und die section2.bin. Insgesamt also folgende Liste:

Code:

Kordon:
-alife_l01_escape.ltx
-way_l01_escape.ltx
Müllhalde:
-alife_l02_garbage.ltx
-way_l02_garbage.ltx
Agroprom:
-alife_l03_agroprom.ltx
-way_l03_agroprom.ltx
Agroprom Untergrund:
-alife_l03u_agr_underground.ltx
-way_l03u_agr_underground.ltx
Dunkles Tal:
-alife_l04_darkvalley.ltx
-way_l04_darkvalley.ltx
Labor X-18:
-alife_l04u_labx18.ltx
-way_l04u_labx18.ltx
Bargebiet:
-alife_l05_bar.ltx
-way_l05_bar.ltx
Wildgebiet:
-alife_l06_rostok.ltx
-way_l06_rostok.ltx
Armeelager:
-alife_l07_military.ltx
-way_l07_military.ltx
Jantar:
-alife_l08_yantar.ltx
-way_l08_yantar.ltx
Labor X-16:
-alife_l08u_brainlab.ltx
-way_l08u_brainlab.ltx
Roter Wald:
-alife_l10_radar.ltx
-way_l10_radar.ltx
Hirnschmelzer:
-alife_l10u_bunker.ltx
-way_l10u_bunker.ltx
Pripjat:
-alife_l11_pripyat.ltx
-way_l11_pripyat.ltx
AKW Tschernobyl:
-alife_l12_stancia.ltx
-way_l12_stancia.ltx
AKW Tschernobyl 2:
-alife_l12_stancia_2.ltx
-way_l12_stancia_2.ltx
Geheimes Monolith-Labor:
-alife_l12u_control_monolith.ltx
-way_l12u_control_monolith.ltx
Der Sarkophag:

-alife_l12u_sarcofag.ltx
-way_l12u_sarcofag.ltx
Sonstige Dateien:
-all.ltx
-section2.binAm einfachsten ist es, wenn ihr diese Dateien nun in einen eigenen Ordner kopiert und euch erst einmal nur die Dateien im Hauptordner lasst, die ihr bearbeiten wollt.
Werfen wir doch mal einen ersten kurzen Blick in eine der Dateien. Ich schlage vor, wir wählen Eintrag Nummer 63 aus der alife_l01_escape.ltx:

Code:

```
[63]
; cse_abstract properties
section_name = actor
name = level_prefix_actor_0001
position = -246.726303100586,-24.7932605743408,-134.433868408203
direction = 0.00431653670966625,-1.39625442028046,-0.00068671052576974
s_flags = 0x29

; cse_alife_object properties
game_vertex_id = 4
distance = 0.699999988079071
level_vertex_id = 11713
object_flags = 0xffffffbf
custom_data = <<END
[dont_spawn_character_supplies]

[spawn]
wpn_binoc
detector_simple
novice_outfit
device_torch
END

; cse_visual properties
visual_name = actors\hero\stalker_novice

; cse_alife_creature_abstract properties
g_team = 0
g_squad = 0
g_group = 0
health = 1
dynamic_out_restrictions =
dynamic_in_restrictions =

upd:health = 1
upd:timestamp = 0x75732029
upd:creature_flags = 0x70
upd:position = -246.726303100586,-24.7932605743408,-134.433868408203
upd:o_model = 0
upd:o_torso = -1.39625442028046,0.00431653670966625,0
upd:g_team = 0
upd:g_squad = 0
```

upd:g_group = 0

; cse_alife_trader_abstract properties
money = 40
trader_flags = 0
character_profile = actor

; cse_ph_skeleton properties

; cse_alife_creature_actor properties

upd:actor_state = 0xd20
upd:actor_accel_header = 0
upd:actor_accel_data = 0
upd:actor_velocity_header = 0
upd:actor_velocity_data = 0
upd:actor_radiation = 0
upd:actor_weapon = 110
upd:num_items = 0Das, was diesen Eintrag so interessant macht, ist die Tatsache, dass dies der Actor-Spawn, also der Spawn des Spielers zu Beginn des Spiels ist.
Ich kann nicht alle der Einträge erklären, da ich sie selbst nicht verstehe, aber es gibt ein paar grundlegende, einfach zu verstehende Einträge. "section_name" ist der Name des Objektes. Hier ist es "actor", also der Spieler, es könnte aber z. B. auch eine Waffe wie "wpn_lr300" sein. "name" ist ein eindeutiger Bezeichner des gespawnten Objektes. Der Unterschied zum "section_name" liegt darin, dass "section_name" praktisch nur die Blaupause ist, nach der dann das eigentliche Objekt "name" erstellt wird. "position" gibt die Position des gespawnten Objektes an, "direction" die Ausrichtung desselben. e wichtige Werte sind die "game_vertex_id" sowie die "level_vertex_id". Dazu gleich mehr.
"visual_name" verweist auf das Model des Objektes, im Falle eines LR300 stünde hier also "weapons\lr300\wpn_lr300". Eine Besonderheit des Actor-Spawns ist die Abteilung "[spawn]" - hier wird die Ausrüstung des Spielers zu Beginn festgelegt. Bei praktisch allen anderen Figuren im Spiel passiert dies in der zu dem entsprechenden Level gehörenden gameplay-XML. Der Aufbau der beiden Dateien ist übrigens recht ähnlich.
Ein kleiner Hinweis noch: Wollt ihr den Spawn des Actors in ein anderes Level verlegen, müsst ihr den Eintrag 63 nicht in die entsprechende Datei kopieren - die korrekte Angabe von position, game_vertex_id und level_vertex_id stellt sicher, dass euer Actor immer dort startet, woh ihr ihn haben wollt, auch in anderen Levels als Kordon.

Kommen wir nun noch einmal zur Position und der gvi bzw lvi. Die Psotion kann man über den Konsolenbefehl "rs_stats 1" im Spiel herausfinden, allerdings nicht so genau wie die Werte in der all.spawn. gvi und lvi lassen sich jedoch nicht so einfach finden - zum Spawnen von Objekten sind sie dennoch nötig. Um diese Werte zu ermitteln, benötigen wir ein kleines Script.
Zunächst einmal müsst ihr die Datei ui_main_menu.script aus den gamedata-Archiven extrahieren(sie liegt im Ordner gamedata/scripts). Öffnet sie und scrollt bis ganz ans Ende. ie letzte Funktion der Datei sollte "function main_menu:OnKeyboard(dik, keyboard_action)" heißen. Ersetzt nun diese Funktion durch die folgenden zwei Funktionen:

Code:
```
function main_menu:OnKeyboard(dik, keyboard_action)  --virtual function
	CUIScriptWnd.OnKeyboard(self,dik,keyboard_action)
	local bind = dik_to_bind(dik)
	local console = get_console()
```

```
	if keyboard_action == ui_events.WINDOW_KEY_PRESSED then
		if dik == DIK_keys.DIK_ESCAPE then
			if level.present() and (db.actor ~= nil) and db.actor:alive() then
				console:execute("main_menu off")
			end
		end

--		if dik == DIK_keys.DIK_S then
--			self:OnButton_load_spawn()

--		else
		if dik == DIK_keys.DIK_Q then
			self:OnMessageQuitWin()
		end

		if dik == DIK_keys.DIK_F10 then
			coordinates_to_message()
		end
	end

	return true
end

function coordinates_to_message()
	-- Put a message about our location on screen
	local a    = db.actor:position()        -- Our position's coordinates
	local lvid  = db.actor:level_vertex_id()
	local gvid = db.actor:game_vertex_id()
	local text = "Position:\\nX= "..a.x.."\\nY= "..a.y.."\\nZ= "..a.z.."\\nlevel_vertex=
"..lvid.."\\ngame_vertex_id= "..gvid
	news_manager.send_tip(db.actor, text, nil, nil, 30000)
	local text =
level.name().."#XYZ:"..tostring(a.x)..","..tostring(a.y)..","..tostring(a.z).."#Lvid:"..tostring(lvid).."#
Gvid:"..tostring(gvid)
	get_console():execute(text)
endDie rot markierten Stücke Scriptcodes sollten jene sein, die bei euch vorher nicht vorhanden waren. Sie bewirken folgendes: Wenn ihr nun im Spiel in das Hauptmenü geht und den Button F10 auf eurer Tastatur drückt, werden eure Position, der aktuelle gvi sowie die lvi ermittelt und als Nachricht auf dem Bildschirm ausgegeben. Da es jedoch recht mühselig wäre, diese einzeln abzuschreiben, werden die Koordinaten auch in eurer Logdatei gespeichert - von dort müsst ihr sie nur noch in die entsprechenden Variablen der all.spawn kopieren. Sehr angenehm.

Also gut, probieren wir mal etwas aus. Startet S.T.A.L.K.E.R. und ein neues Spiel. Wartet, bis Sidorowtisch euch aus seinem Bunker entlässt, sucht euch ein schönes Plätzchen im Dorf, geht ins Menü und drückt F10 - ein leiser Ton sollte den Empfang eurer Nachricht bestätigen. Bei mir sah es so aus:

Verlasst das Spiel nun wieder und öffnet sowohl die alife_l01_escape.ltx als auch eure Logdatei. Dort solltet ihr einen ähnlichen Eintrag wie den meinen finden:

Code:
```

! Unknown command: l01_escape#XYZ:-220.43411254883,-18.416578292847,-139.12705993652#Lvid:33469#Gvid:64Dort haben wir schon den Großteil der Informationenn, die wir für einen erfolgreichen Spawn brauchen. Geht nun ans Ende der alife_l01_escape.ltx und legt einen neuen Eintrag an. Wichtig dabei ist, dass euer Eintrag nicht die Nummer eines bereits bestehenden Eintrages haben darf. S.T.A.L.K.E.R. hat um die 8400 oder 8500 Einträge, daher beginne ich den neuen Eintrag testweise einmal mit der Nummer 9000:

Code:
[9000]
; cse_abstract properties
section_name = wpn_lr300
name = esc_wpn_lr_0000
position = -220.43411254883,-18.416578292847,-139.12705993652
direction = 0,-1,-1

; cse_alife_object properties
game_vertex_id = 64
distance = 0
level_vertex_id = 33469
object_flags = 0xffffff0f

; cse_visual properties
visual_name = weapons\lr300\wpn_lr300

; cse_alife_item properties
condition = 0.999999582767487

upd:num_items = 0

; cse_alife_item_weapon properties
ammo_current = 90
addon_flags = 2

upd:condition = 255
upd:weapon_flags = 0
upd:ammo_elapsed = 0
upd:addon_flags = 2
upd:ammo_type = 0
upd:weapon_state = 0
upd:weapon_zoom = 0

upd:current_fire_mode = 0

upd:grenade_mode = 0Der Eintrag wurde von dem Spawn eines LR300 im Armeelager übernommen und dahingehend modifiziert, dass er auf der Mauer von meinem Screenshot oben ein LR300 spawnt. Wenn ihr wollt, könnt ihr die Position und die gvi und lvi durch eure eigenen Werte ersetzen. Ihr könntet auch, wenn ihr Lust darauf habt, ein wenig mit den Werten für die direction herumspielen.

Sobald ihr damit fertig seid, werden wir die Dateien wieder zu einer einzigen all.spawn zusammenkompilieren. Kopiert, sofern ihr sie in einen anderen Ordner verschoben habt, wieder alle Dateien in den Hauptordner von ACDC(natürlich ohne dabei die modifizierten Dateien zu

überschreiben) und startet die compile.bat. Es dauert einen kurzen Moment, bis das Fenster verschwindet und euch eine Datei namens new.spawn in eurem Ordner hinterlässt. Diese kopiert ihr nun in den Ordner gamedata/spawns in eurem S.T.A.L.K.E.R.-Hauptverzeichnis und benennt ihn um zur all.spawn. Wenn ihr nun ein neues Spiel startet, erwartet euch an der vorgegebenen Position auch schon euer nagelneues LR300:

Übrigens: Damit ihr Munition in der gespawnten Waffe habt, müsst ihr den Wert "upd:ammo_elapsed = 0" ändern.

Zum Spawnen von Objekten war dies im Grunde auch schon das wichtigste. Probiert mal ein wenig rum und ersetzt z. B. herumliegende Waffen durch andere. Wie wäre es zum Beispiel damit, wenn die Groza hinter Sidorowitsch durch ein AK-74 ersetzt werden würde? Oder wenn im Militäraußenposten statt einem AKS-74u ein FN F2000 läge?

Ein kleines Detail möchte ich aber noch erwähnen: Neue Gegenstände können nicht einfach gespawnt werden, sondern müssen ACDC quasi "bekannt sein."
Doch gerade das Spawnen neuer Gegenstände ist wohl - mit einem Blick auf die Arsenal-Mod - ein sehr interessanter Punkt. Daher möchte ich noch eben erwähnen, wie man zumindest eine neue Waffe zu ACDC hinzufügt:
Öffnet die acdc.pl-Datei - dazu könnt ihr jeden beliebigen Texteditor nehmen. Sucht nun nach den Einträgen für die Waffen - glücklicherweise beginnen sie alle mit "wpn_" und lassen sich so leicht finden.
Was ihr findet, sieht z. B. so aus:

Code:
wpn_mp5 => 'cse_alife_item_weapon_magazined',
wpn_mp5_m1 => 'cse_alife_item_weapon_magazined',
wpn_mp5_m2 => 'cse_alife_item_weapon_magazined',Wollt ihr nun eine neue Waffe spawnen, müsst ihr genau solch einen Eintrag den alten Einträgen hinzufügen. Beachtet dabei jedoch unbedingt, dass ihr der neuen Waffe auch den richtigen Typ zuweist - vertut ihr euch hier, können sehr merkwürdige Sachen geschehen. Ein AK-108 ist z. B. - da man einen Granatwerfer anbringen kann - kein Objekt des Typs "cse_alife_item_weapon_magazined", sondern eines des Typs "cse_alife_item_weapon_magazined_w_gl"! Als ich diesen Unterschied einmal übersah, hielt ich ein AK-108 mit mehr als 12000 Schuss und einem miserablen Zustand in den Händen!
Wolltet ihr nun z. B. ein HK417 spawnen, müsstet ihr voher also noch folgenden Eintrag in acdc.pl ergänzen:

Code:
wpn_hk417_sk1 =>
'cse_alife_item_weapon_magazined_w_gl',Glücklicherweise ist die Datei schön einfach aufgebaut und so fällt diese Arbeit auch nicht sonderlich schwer

Erläuterung der Munitionsparameter

--------by Krishty

$spawn - Name im Level Editor: "weapons\ammo\ammo_example"
visual - Name und Pfad des Models - weapons\ammo\ammo_example.ogf
box_size - Anzahl Patronen pro Paket.
cost - Preis pro Paket, in Rubel.
inv_name - Verweist auf den String mit dem vollen Namen (string_table_enc_weapons.xml): ammo-example
inv_name_short - Verweist auf den String mit dem kurzen Namen (string_table_enc_weapons.xml): ammo-example_s. Sichtbar z.B. neben dem Waffennamen rechts unten im HUD, also nicht zu lang machen.
description - Name des Strings in der string_table_enc_weapons.xml, der die Beschreibung des Objekts enthält: enc_weapons1_ammo_ammo-example
inv_weight - Gewicht, in Kilo.
inv_grid_width - Siehe Erklärungen zu Inventar-Icons...
inv_grid_height - "
inv_grid_x - "
inv_grid_y - "
buck_shot - Anzahl der Schüsse, die pro Patrone gelöst werden. Mehrere bei Schrot, sonst immer nur einer.
k_disp - Koeffizient für die Streuung der Waffe bei dieser Munitionsart aus der Waffenname.ltx. Siehe fire_dispersion_base in den Erklärungen der Waffenparameter.
k_dist - Koeffizient für die Reichweite der Waffe bei dieser Munitionsart aus der Waffenname.ltx. Siehe fire_distance in den Erklärungen der Waffenparameter. Hat keinen Einfluss auf die Schussbahn.
k_air_resistance - (Nur im Multiplayer verwendet!) Koeffizient für den Luftwiderstand der Patrone.
k_hit - Koeffizient für die Trefferwirkung aus der Waffenname.ltx (hit_power). Wird durch viele weitere Faktoren beeinflusst, z.B. das getroffene Körperteil, die Distanz zum Schützen, etc. pp.
k_pierce - Die Penetrationswirkung der Waffe. 1.15 reicht zum Durchschießen einer Holztür, 1.35 zum Durchschießen zweier, 1.4 für drei. Hat keinen Einfluss auf Anzüge!
k_ap - Wieviel der Verletzungsstärke hinter Anzüge transportiert wird. Im Singleplayer-Modus wirkt zusätzlich immer ein gewisser Prozentsatz hinter der Weste, rund ein Achtel. Das lässt sich nicht abschalten und sollte in diesen Wert miteinkalkuliert werden.
k_impulse - Koeffizient für die Impulsübertragung aus der Waffenname.ltx (hit_impule).
impair - Koeffizient für die Verschmutzung der Waffe pro Schuss (condition_shot_dec).
tracer - (Kosmetisch) Leuchtspur? (on/off) Achtung: Falls in einer Waffen-Section tracers definiert wird, tritt dieser Wert außer Kraft!
tracer_color_ID - (Kosmetisch) Index der Leuchtspurfarbe im tracers_color_table des bullet_manager der weapons.ltx. Achtung: Falls in einer Waffen-Section tracers_color_ID definiert wird, tritt dieser Wert außer Kraft!
wm_size - (Kosmetisch) Seitenlänge der Einschusslöcher, in Meter. Die Texturen sind meist vier- oder fünfmal so groß wie die eigentlichen Einschusslöcher, damit drumherum z.B. der abgeplatzte Putz erscheint. Das sollte miteinkalkuliert werden.
can_be_unlimited - Wahrscheinlich unbenutzt und Schlacke aus einer früheren Engine-Version.
explosive - (Kosmetisch) Explosions-Animation beim Einschlag? (on/off)

Programme für das Modden mit S.T.A.L.K.E.R.

Das Ganze soll eine Zusammenstellung der Programme die für das modden von S.T.A.L.K.E.R. benötigt sein. Damit solltet ihr dann in den Lage Neue Icons in Stalker zu bringen oder die Levels zu bearbeiten, falls euch irgendein Programm bzw. Plugin für z.b. 3DS MAX fehlt werdet ihr es

hier bestimmt finden.

1.Grafik-Bearbeitung

Paint.Net: http://www.chip.de/downloads/Paint.NET_13015268.html
Gimp: http://www.chip.de/downloads/GIMP_12992070.html

2.Text-Bearbeitung

Notepad: http://www.chip.de/downloads/Notepad_12996935.html

3.Spawn-Bearbeitung

ACDC-Tool: http://www.stalkerin.gameru.net/deve.../acdc11oct.rar
XrSpawner: http://rapidshare.com/files/75053416...1Beta_engs.rar

4.Level-Bearbeitung

SDK: http://files.gsc-game.com/st/xray-sdk-setup-v0.4.exe
AI-Wrapper: http://stalkerin.gameru.net/modules....heoned&lid=166
AI-Compiler: http://stalkerin.gameru.net/modules....heoned&lid=165

5.Tools und Plugins

3DS MAX .geom Plugin: http://stalkerin.gameru.net/modules....heoned&lid=169
3DS MAX .object Plugin: http://forum1.onlinewelten.com/redir...2F0-0-0-146-20
3DS MAX .smd Plugin: http://forum1.onlinewelten.com/redir..._max5.013a.rar
UI Merge Tool (ui_icons_equiqment): http://www.smart-mod-manager.org/other/ui-merge-0.4.7z
Smart Mod Tool: http://www.codeplex.com/Release/Proj...eleaseId=10596
Smart Terrain Debug and Waypoint Extractor Tool: http://sdk.stalker-game.com/en/image...y_dez0wave.zip
LUA Compiler: http://www.lua.org/
ogf2smd: http://forum1.onlinewelten.com/redir...it%26lid%3D118

Als Ergänzung zu den FAQs von nihilant und DerSchwabe, möchte ich noch die folgenden Befehle kurz erklären:

Code:
```
<dialog id="xxx">
<precondition>dialog_manager.precondition_is_phrase_disabled</precondition>
      <phrase_list>
        <phrase id="0">
          <text>xxx_0</text>
          <action>dialog_manager.action_disable_phrase</action>
          <next>1</next>
        </phrase>
          <phrase id="1">
          <text>xxx_2</text>
          <next>2</next>
        </phrase>
```

...Diese Befehle bewirken den Effekt, den man im Spiel bei fast allen Charakteren beobachten kann: Man spricht sie auf einen Dialog an und wenn dieser abgeschlossen ist, erscheint er solange nicht, bis man entweder das Level neu betritt oder einen Spielstand lädt.
Der Befehl:

Code:
<action>dialog_manager.action_disable_phrase</action>sollte übrigens immer in der phrase mit der id="0" stehen. Der

Code:
<precondition>dialog_manager.precondition_is_phrase_disabled</precondition>sollte immer vor der phrase_list stehen!

Ein weiterer Befehl ist folgender:

Code:
<phrase id="xxx">
<text>...</text>
<action>dialogs.break_dialog</action>
</phrase>Dieser Befehl bewirkt, dass der Dialog mit dem Charakter sofort beendet wird. Kann man beispielsweise nutzen, wenn man einen Charakter verärgert hat oder er einem nichts mehr zu sagen hat.
Wichtig dabei ist, dass der:

Code:
<action>dialogs.break_dialog</action>Befehl immer in einer Sprecheinheit kommt, in der der Spieler etwas sagen würde! Wenn man also die Gesprächseinheiten aufzählt:
1. Spieler
2. NPC
3. Spieler
4. NPC usw.
Dann sollte dieser Befehl nicht bei einer geraden Nummer kommen. Wieso?
Nun, man stelle sich folgendes Gespräch vor:
S: Hi!
NPC: Hi!
S: Wie gehts?
NPC: Gut!
Und würde in der Dialog.xml in der 4. <phrase>...</phrase> den Befehl noch zusätzlich setzen, würde das Gespräch so aussehen:
S: Hi!
NPC: Hi!
S: Wie gehts?

Um dies zu verhindern, sollte man noch eine 5. phrase mit beispielsweise ... dran setzen, wie ich es oben zu Beginn gemacht habe:

Code:
<phrase id="xxx">

<text>...</text>
<action>dialogs.break_dialog</action>
</phrase>

Wenn man dem Spieler mehrere Antworten zur Auswahl geben möchte, muss man einfach nur an der entsprechenden Stelle mehrere <next>...</next> Befehle geben:

Code:
...
<phrase id="1">
<text>xxx_1</text>
<next>2</next>
<next>3</next>
</phrase>Man sollte dabei aber immer drauf achten, wer ab welcher stelle was sagt, damit man nicht mit Sprecher und Zuhörer durcheinander kommt

Auch wenn es schon erwähnt ist, kann man bestimmte Gesprächsinhalte erst sichtbar machen, wenn man info_portions gibt.
Mit den Befehlen:

Code:
<has_info>...</has_info>

und

<dont_has_info>...</dont_has_info>können Bedingungen gesetzt werden, ab wann eine Gesprächsoption sichtbar wird. Hier dazu noch 2 Beispiele, wo überall diese gesetzt werden können:

Code:
...
<phrase id="1">
    <text>xxx_1</text>
    <next>2</next>
    <has_info>...</has_info>
    <next>3</next>
</phrase>Bewirkt, dass auf phrase 3 erst verlinkt wird, wenn die entsprechende info_portion gegeben wurde.

Alternativ und besser ist es jedoch es so zu machen:

Code:
...
<phrase id="1">
    <text>xxx_1</text>
    <next>2</next>
    <next>3</next>

</phrase>
...
<phrase id="3">
<has_info>...</has_info>
<text>xxx_3</text>Bewirkt das gleiche und ist etwas übersichtlicher.

Code:
<dialog id="aaa">
<dont_has_info>xxx</dont_has_info>
<has_info>yyy</has_info>
<phrase_list>
...Bewirkt, dass das ganze Gespräch nur dann verfügbar ist, wenn der Spieler die info_portion xxx nicht hat, dafür aber schon die yyy.

NEU:

Code:
<start_dialog>xyz_start_dialog</start_dialog>Sollte so ein Eintrag in einer Charcter_descr_... stehen, bedeutet dies, dass der genannte Dialog immer als erstes vor allen anderen Dialogen aufgerufen wird. Dies ist vor allem in CoP und CS der Fall, für SoC weiß ich es grad nicht.

Dort kann man dann folgendes beobachten in der entsprechenden Dialog Datei:

Zitat:
<phrase_list>
<phrase id="0">
<text />
<next>1</next>
</phrase>
<phrase id="1">
<text />
<next>11</next>
<next>12</next>
<next>13</next>
</phrase>
...

Der Befehl <text /> überspringt hier den Text und es wird keiner angezeigt. Dadurch ist in der Reihenfolge der NPC an der Reihe und nicht der Spieler.
Gleiches geschieht dann in phrase id="1". Dies muss auch so sein, ansonsten würde der Spieler an der Reihe sein. Ist etwas kompliziert zu erklären:
Die Reihenfolge ist immer
Spieler, NPC, Spieler, NPC, ...

Ist der Spieler an der Reihe, bleibt er es solange bis
1. er eine <text> Passage </text> gesagt hat
2. Oder er auf eine leere <text /> Passage verweist.

In unserem Fall wäre in phrase id="1" der NPC dran, da dort eine leere <text /> Passage steht. Das ist richtig, wegen Punkt 2.
Diese Phrase verweist auf drei andere Phrases mit Text und dadurch gehören diese drei zu dem

NPC, der gerade dran ist.
Ist etwas schwer zu verstehen, aber braucht man auch sehr selten. Einfach an das vorhandene Muster halten und immer ausprobieren.

Der NPC ist also immer noch an der Reihe und hat hier drei Gesprächsoptionen zur Auswahl. Wenn diese nicht an Bedingungen geknüpft sind, wird zufällig einer der drei Dialoge ausgewählt. Dies ist zum Beispiel im Spiel bei den Barmännern zu sehen, denn diese Begrüßen den Spieler immer zufällig mit einer der drei Begrüßungen.

Welchen Wert muss ich bei "scope_zoom_factor" für
einen bestimmten Vergrößerungsfaktor eingeben?

Der Wert "scope_zoom_factor" bestimmt die Vergrößerung von Zielfernrohren.
Leider gibt er nicht den Zoomfaktor, sondern nur den Winkel des Gesichtsfeldes an. Will man also einen bestimmten Faktor erhalten, muss man eine Beziehung zwischen ihm und dem Wert "scope_zoom_factor" herstellen.
Dazu braucht man den Winkel, den das Gesichtsfeld ohne den Blick durch ein Zielfernrohr hat. Bei SoC beträgt er 90°. Das hört sich vielleicht nicht nach viel an, ist aber deutlich mehr, als z. B. in Modern Warfare 2, wo er nur magere 65° hat.

Die 90° werden folglich durch eine Zahl x dividiert, woraus der Wert "scope_zoom_factor" entsteht. Diese Zahl x verhält sich zur gewünschten Vergrößerung wie 3 zu 2. Oder anders ausgedrückt: Man muss 90° durch den Wert teilen, der 1,5 mal so groß ist, wie der Zoomfaktor, den man am Ende haben will.

Hier einige Beispiele:

1,2x entspricht 90 / 1,8 = 50 denn 1,8 / 1,2 = 3 / 2

1,5x entspricht 90 / 2,25 = 40 denn 2,25 / 1,5 = 3 / 2

2x entspricht 90 / 3 = 30 denn 3 / 2 = 3 / 2

3x entspricht 90 / 4,5 = 20 denn 4,5 / 3 = 3 / 2

4x entspricht 90 / 6 = 15 denn 6 / 4 = 3 / 2

6x entspricht 90 / 9 = 10 denn 9 / 6 = 3 / 2

8x entspricht 90 / 12 = 7,5 denn 12 / 8 = 3 / 2

Die unterstrichenen Werte müssen bei "scope_zoom_factor" eingetragen werden.