

Universität von PARAIISO

Armed Manual V 2.04

geschrieben von vielen ArMA Fans.

Vorwort:

Die im Dezember 2006 gegründete Universität von Paraiso ist ein Forschungsinstitut welches sich dem Ziel verpflichtet hat Grundlagen und Errungenschaften in der ArMA Welt zu erforschen.

In diesem Manual werden die Grundlagen sowie der Umgang mit dem Editor erklärt, sowie einige Beispiele in den Bereichen Skripting und Missionsdesign. Wie alle Forschungsstationen publiziert auch die Universität von Paraiso ihre Forschungsergebnisse. Es wird versucht dieses Manual auf dem Stand der jeweiligen ArMA Version zu halten. Dieses Werk ist für Version 1.07 geschrieben.

Weitergehend sollte jedem klar sein das eine gute Mission nicht in ein paar Minuten geschrieben wird. Es ist ratsam den Missionsablauf zuerst auf Papier zu skizzieren. Erst wenn ein guter Plan steht, wird die Mission auch rund... So, nun viel Spass beim editieren.



Inhaltsverzeichnis

<i>Optionen Spieleinstellungen</i>	9
Audio-Optionen	9
Video-Optionen	10
Steuerung konfigurieren	11
Schwierigkeitsgrad	12
Rasenmäher	13
<i>Allgemeine Hilfe in ArMA</i>	14
Armed Assault – Parameter	14
Video-Optionen	15
Modifikationen	15
Einfügen von Custom Faces	18
Einfügen von Custom Sounds	18
<i>Starten des Editors</i>	19
Der Editor	20
Einstellungen im Editor.	20
Die Tasten F1 – F6	22
Einheit Hinzufügen F1	22
Gruppe Hinzufügen F2	24
Auslöser hinzufügen F3	25
Effekte im Auslöser	27
Wegpunkte hinzufügen F4.....	28
Synchronisieren F5	30
Markierung hinzufügen F6	31
<i>Editorbeispiele</i>	32
Einheit erstellen	32
Gruppenerstellen	33
Kennzeichnungen der Einheiten	34
Wegpunkt	34
Einheit unter einer Wegmarke	36
Synchronisieren	36
Auslöser	37
Mission weiter bearbeiten	37
Mission beenden.	38
Schleifen laufen (patrouillieren)	39
Einheit geht in ein Gebäude	40
Helikopter	41
Helikoptertransport Abladen	41
Helikoptertransport Abholen	42
Helikopterlandung	43

Helikopter Abladen mal anders	43
Helikopter Einsammeln mal anders	44
<i>Editieren</i>	45
Einheit beschädigen	45
Entfernung zweier Objekte / Units abfragen	45
Auslöser versetzen	45
Aktion im Aktionsmenü eintragen	45
Fahne am Mast	45
Schranke öffnen – schliessen	45
Objekte / Einheiten schweben lassen	46
Erstellen neuer Einheiten	46
Einheit löschen	46
Sichtweite im Spiel anpassen	46
Zeitmaschine	46
Marker sichtbar machen	47
Inselkarte auf den Monitor bringen	47
Eingaben des Spielers abschalten	47
Bestimmten MP player ansprechen	47
Einheiten in einem Bereich ansprechen	47
Feindgebiet ist Feindfrei	48
Todeszone	48
Einheiten in einem Bereich zählen	48
Möglichkeit über eine Liste	48
Objekte schräg platzieren	49
Objekte auf der Achse drehen	49
Objekte auf der Achse kippen	49
<i>Speichern</i>	50
Mission speichern	50
Missiondaten erfassen / speichern	50
Playerstatus über ID erfassen	50
Variablen Speichern	50
Speicher löschen	51
<i>Funkverkehr</i>	51
Funksprüche	51
Gruppe ein Rufzeichen geben	51
Textmitteilungen	51
<i>AI / KI System</i>	52
Einheit plazieren	52
Gruppe Starten / Stoppen	52
Körperhaltung der Einheit	52
Animation einer Einheit	52

Blickrichtung der Einheit	53
Blickrichtung aufheben.....	53
Einheit feuert auf Einheit	53
Gesichtsausdruck	53
Einheiten in Vehicle	53
Gruppe im Laderaum eines LKWs / Helikopters / Boot etc.....	54
Bestimmte Einheit in Vehicle ansprechen.....	54
Bestimmte Einheit in Vehicle löschen.....	54
Einheit am Fallschirm.....	54
Nur bestimmter Typ darf die Einheit Nutzen.....	55
Vehicle beschleunigen	55
Verhalten, Geschwindigkeit einer Gruppe.....	55
Verhalten einer Einheit.....	55
Feiger Feind	56
Formation.....	56
Einheiten formieren	57
Waffen an die AI verteilen.....	57
Vehicle bewaffnen	58
Prominente Einheiten.....	58
Alarm abspielen.....	58
Alarm auslösen	59
Waffen & Munition der Einheit erfassen.....	60
Tiefflug	60
Einheiten zu einer anderen Einheit gehen lassen.....	61
Oder zu einem Objekt.....	61
Der Feind in meinem Bett.....	61
Feindlicher Kamerad	61
Gegner freundlich	61
Gegner wieder feindlich	61
Suchscheinwerfer ein und aus schalten	62
Spielbare Einheit hinzufügen	62
<i>Der MissionsOrdner.....</i>	63
Missions.sqm.....	64
Description.ext	68
Klassendeklarationen.....	68
Identität festlegen	69
Mission freischalten.....	69
Typ der Mission.....	69
Missionsname	69
Zeitanzeige	70
Ausrüstung für den Spieler	70

Punkte für den Spieler	70
Waffenwahl im Briefing	71
Beispiel einer description.ext einer Multiplayer Mission	71
Beispiel : description.ext	72
Elegante Lösung	74
Textersatz	75
KI in der Abschussliste aufführen	75
Briefing	75
Hinzufügen weiterer Missionsziele:	76
Hinzufügen weiterer Seiten	76
Die HTML Hauptseite	77
Name des HTML Briefings:	77
Missionsziele abhaken:	77
Die Overview	78
Stringtable.csv	79
<i>Arbeiten mit Skripten</i>	<i>80</i>
Eine Übersicht in Beispielen	80
Arrays	80
BOOLSche Befehle	80
Berechenbarkeit von Variablen	81
Variablen	81
Erste Schritte	81
Starten eines Skripts aus dem Editor	82
Starten eines Skripts aus einem Skript	82
Starten eines Skripts aus einer Funktion	82
Starten einer Funktion aus einem Skript	82
Die Arithmetik	82
Anweisungen	82
Pause	83
Pause bis Bedingung erfüllt ist	83
Vergleiche	83
Vorrang und Assoziativität der Operatoren	83
Sprung Marker	84
Schleifen	84
Ausgaben auf den Bildschirm	84
Auslesen von Parametern	84
Erklärungen hinterlassen	85
Abfragen von Werten	85
Setzen von Werten	85
Ausgabe von Werten	85
Scriptbeispiele:	85

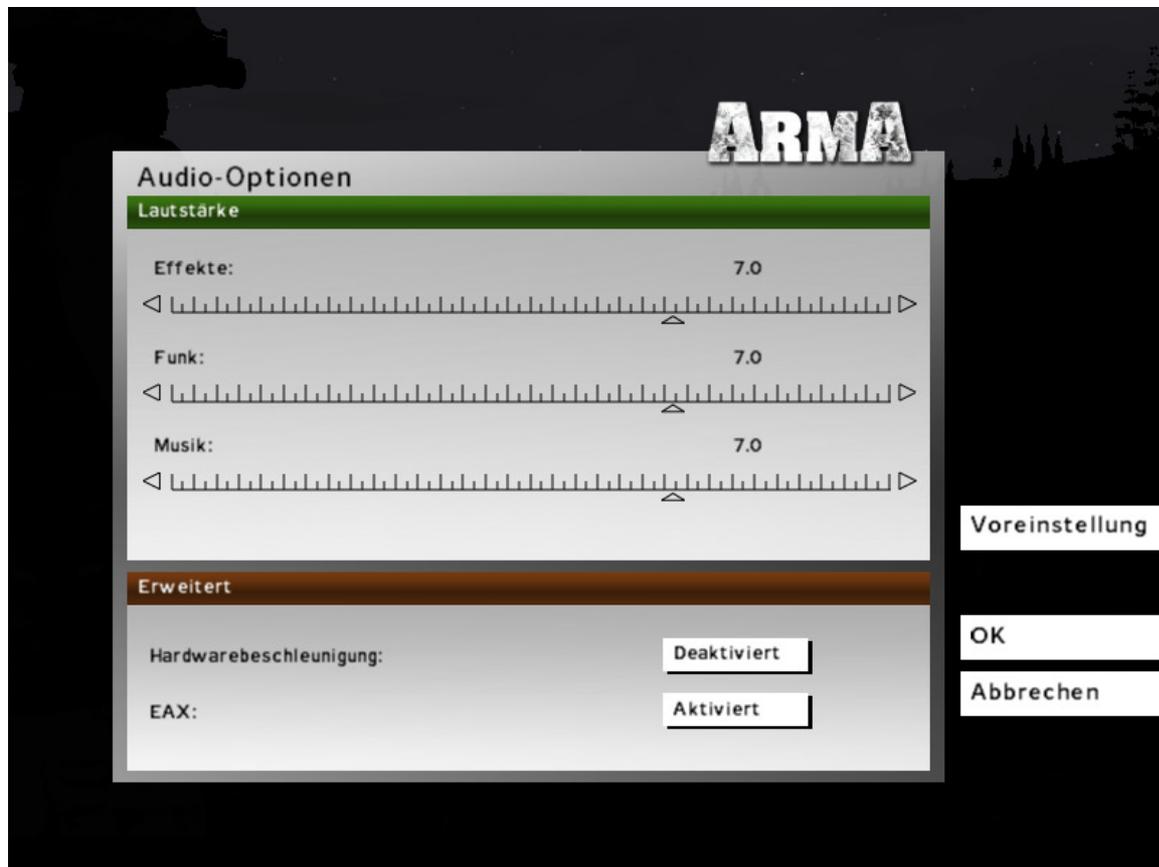
Unterstützung Rufen.....	86
Speed	86
Das FallschirmSkript	87
Das GPS Marker Skript1	88
Das GPS Marker Skript2	88
Das GPS Marker Skript3	89
Evakuierungs Punkt aussuchen.....	89
Artillerie	90
Artillerie1	91
Artillerie mit Geschützen.....	92
Kamera Camera	92
Digitaler Zoom	93
Musik starten	94
Musik beenden.....	94
Kamera im Gelände fixieren.....	94
Variablen setzen	94
Kamera über einem Objekt Erstellen.....	94
Kamera an eine Position beamen.....	94
Kamera an ein Objekt heften.	95
Methode zwei	95
Kamera Effekte.....	95
Beispiel einer Kamera Führung	96
Die Lippen	96
Camera.sqs.....	97
Cameraflug aufzeichnen	97
MP Skripte	98
Die glorreichen Vier	98
Arbeiten mit Funktionen.....	99
Funktionen	99
Funktion starten	99
Anweisungen und Blöcke	99
Erklärungen hinterlassen	99
Pause.....	99
Werte zuweisen	99
Array einlesen auslesen.....	99
if-else.....	100
Schleifen	100
while	100
for	101
if-else und Schleifen verschachteln	101
Schleifen verschachteln.....	102

Das Fallschirmspringer Skript als Funktion.....	102
Das GPS Skript als Funktion.....	103
Wettersystem	103
<i>Funktionen entwickeln</i>	<i>106</i>
 Zeiger und Vektoren	106
 Zeiger und Adressen	106
 Zeiger und Funktionsargumente.....	107
 Distanz XY und XYZ	108
 Erste eigene Funktion	109
<i>Effekte.....</i>	<i>110</i>
 Feuer in der Mission.....	110
 Rauchgranaten und Flare erzeugen.....	110
 Sprengungen erzeugen auslösen.....	111
 Brennendes explodierendes Fass.....	111
Feuer breitet sich aus	112
 Partikel erzeugen (Drop Array).....	112
Es schneit.....	115
Erstellen einer Rauchsäule.....	116
Noch eine Rauchsäule	116
Erstellen einer Feuer Rauchsäule.....	117
Feuer muss bei Nacht leuchten!.....	118
Kamin in Aktion	119
Feuer im Triebwerk	119
 Erstellung von Lichtquellen	120
 Fliegen an der Leiche	122
<i>Die Tierwelt</i>	<i>123</i>
<i>Dialoge.....</i>	<i>127</i>
 Button & co.....	129
Statischer Text feste Texte ohne Funktion CT_STATIC.....	129
Aktiver Text Bedienbarer Text CT_ACTIVETEXT	130
Buttons Knöpfe CT_BUTTON.....	130
Textfelder Eingabefelder CT_EDIT	130
ListBoxen CT_LISTBOX.....	130
ComboBoxen CT_COMBO	130
<i>Multiplayer</i>	<i>131</i>
 Respawn	131
 Multiplayer und Server Installation	132
 Beispiel Config.....	134
 XML	135
 Netzwerkoptionen.....	136
 Die Koordinaten	136

NATO-Alphabet	137
<i>Insel erstellen</i>	137
Die Insel PBO.....	137
Der Inselordner	137
Config.cpp	138
Das WRP Format	143
<i>DatenBank</i>	144
Westwaffen und Ausrüstung	144
Ostwaffen und Ausrüstung.....	148
Ausrüstung Fahrzeuge	150
Skript Befehle	160
Animationsliste	221
<i>Quellen:</i>	237

Optionen Spieleinstellungen

Audio-Optionen



Lautstärke:

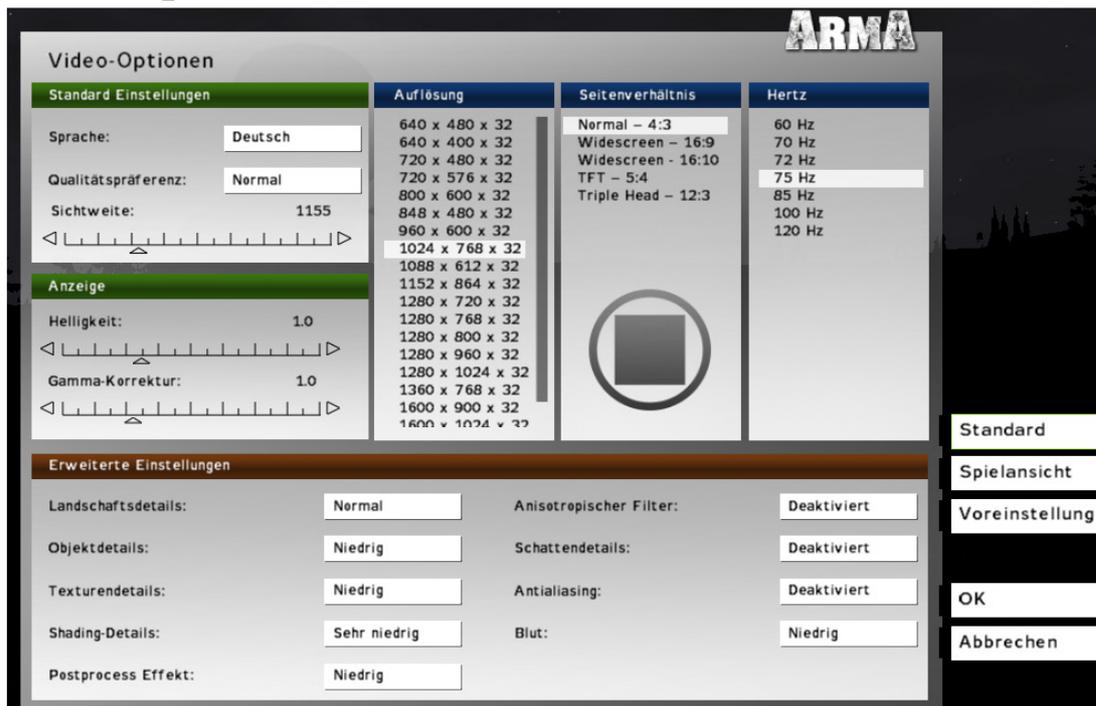
Dem Spieler stehen drei Hauptregler zur Verfügung. Wenn diese verstellt werden dann erfolgt eine Soundeinspielung welche der gewählten im Spiel entspricht. Der erste ist für die im Spiel Auftretenden Effekte. Explosionen Schüsse Motorengeräusche. Der Mittlere (Funk) bestimmt die Lautstärke der Stimme von Funknachrichten und den Stimmen der Einheiten. Der untere regelt die Lautstärke der Musik Einspielungen. (Diese können vom Spieler auch je nach Eintrag verändert werden (Siehe description.ext)

Mit OK werden die Einstellungen übernommen, mit Abbrechen wird der Bildschirm nicht übernommen. Mit Voreinstellung wird die Vorgabe des Spiels geladen.

Hardwarebeschleunigung kann aktiviert oder deaktiviert werden. (Diese Einstellung ist Soundkarten abhängig)

EAX bietet dem Spieler einen simulierten 3D Sound an. Diese Einstellung bewirkt das sich Stimmen (Echos im Gebirge) oder Rotorgeräusche verändern (Hinter Häusern Bergen etc.).

Video-Optionen



Standard Einstellungen:

Sprache: Hier werden die Spracheinstellungen für Missionsbeschreibung, Editor und allgemeine Knöpfe bestimmt. (Neustart erforderlich)

Qualitätspräferenz: allgemeine Haupteinstellung. Regelt alle Einstellungen (Sichtweite, Erweiterte Einstellungen) auf das gewünschte Hauptlevel.

Sichtweite: Dieser Schieberegler bestimmt die dargestellte Sichtweite. Angabe in Metern.

Anzeige Helligkeit: Regelt die Helligkeit auf dem Monitor. (Dies mit dem Monitor abstimmen)

Gamma-Korrektur: Farbwerte des Monitors werden abgestimmt. (Dies mit dem Monitor abstimmen)

Auflösung: Hier wird die Auflösung mit dem das Spiel dargestellt werden soll bestimmt. (Achtet auf die Werte eures Monitors, oder übernehmt eure Einstellungen aus der Windoof Oberfläche)

Seitenverhältnis: Hier wird die Form eures Monitors gewählt. (Achtet auf die Werte eures Monitors. Oder übernehmt eure Einstellungen aus der Windoof Oberfläche)

Hertz: Bestimmt die Pulsanzahl mit der euer Monitor angesprochen wird. (Achtet auf die Werte eures Monitors, oder übernehmt eure Einstellungen aus der Windoof Oberfläche)

Erweiterte Einstellungen: Landschaftsdetails: Hier wird die Darstellung der Landmasse bestimmt. Auflösung der aufgebrachten Texturen bestimmt.

Objektdetails: Bestimmt die LODs (Level of Detail). Hier wird eingestellt wann ein Objekt ganz aufgebaut wird oder abgebaut wird.

Texturdetails: Regelt die Texturen auf den Objekten. (Auflösung)

Shading-Details: Shading ist für 3d Effekte auf z.B. der Uniform zuständig.

Postprocess Effekt: Regelt die verschwommenen Effekt bei Sachen im peripheren Sichtbereich oder weiter weg.

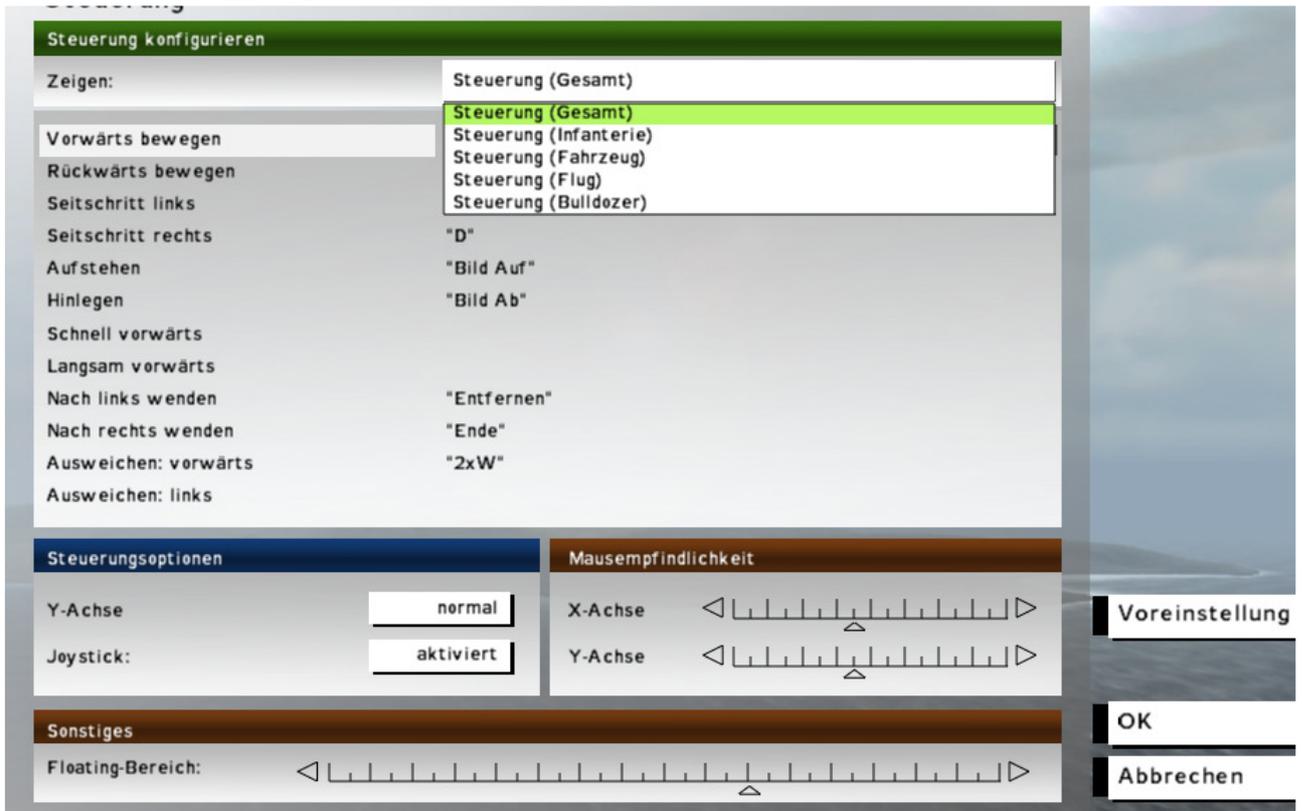
Anisotropischer Filter: Erzeugt weiche Übergänge zwischen Texturen

Schattendetails: Bestimmt die Darstellung der Schatten. Diese können grob bis sehr fein sein.

Antialiasing: glättet Kanten. Sorgt dafür das keine Stufen zu sehn sind.

Blut: Bestimmt die Darstellung von Bluteffekten. (In Zeiten von Killerspielen eine Rarität)

Steuerung konfigurieren



Zeigen: Hier kann eine Auswahl je nach Einheiten Typ erfolgen. Somit kann nur die Infanterie bearbeitet werden oder nur Luft Einstellungen angezeigt werden. Jede Aktion ist hier aufgeführt. Diese kann ausgewählt werden und einer anderen Taste zugeordnet werden. Weitergehend besteht die Möglichkeit Aktionen mehrfach zu belegen oder mit zwei Tasten zusammen zu belegen.

Steuerungsoptionen: Hier wird festgelegt ob ein Joystick eingebunden werden soll und ob dessen Y Achse invertiert oder normal sein soll.

Mausempfindlichkeit: Der X und Y Achse können Empfindlichkeiten zugeordnet werden. Dies bestimmt die Reaktion der Einheit auf eine Mausbewegung.

Floating Bereich: Hier kann der Bereich eingestellt werden in dem sich die Maus Waffe frei bewegen kann bevor sich die Spielfigur anfängt zu drehen.

Mit OK werden die Einstellungen übernommen, mit Abbrechen wird der Bildschirm nicht übernommen. Mit Voreinstellung wird die Vorgabe des Spiels geladen.

Schwierigkeitsgrad



Es können zwei Spielvarianten unabhängig voneinander bestimmt werden. Der Anfängertyp: Kadett und der Expertentyp: Veteran Modus. Einige Einstellungen im Editor (hintc usw.) laufen nur im Kadett Modus. Einige Optionen sind im Veteran Modus bereits fest belegt und können nicht verändert werden.

Verstärkte Panzerung: Erhöht die Panzerung (Im Veteran Modus deaktiviert)

Eigene Markierung: Zeichnet einen Marker auf die Spielfigur. (Im Veteran Modus deaktiviert)

Gegner Markierung: Zeichnet Gegner auf der Karte ein. (Im Veteran Modus deaktiviert)

Erweiterte HUD-Informationen: Zeigt Namen Typ von Gegnern. Markiert die Formationsposition. Blendet diese wieder aus.

Ständig erweiterte HUD-Informationen: Zeigt Namen Typ von Gegnern. Markiert die Formationsposition. Blendet diese nicht wieder aus.

HUD-Wegpunktinfo: Zeigt bei Wegpunkten die Richtung des nächsten an oder gibt Infos über Position. Blendet diese wieder aus.

Ständig HUD-Wegpunktinfo: Zeigt bei Wegpunkten die Richtung des nächsten an oder gibt Infos über Position. Blendet diese nicht wieder aus.

Automatische Meldung: Gibt automatisch Meldung. Position etc.

Erweiterte Karteninformationen: Zeichnet auf der Karte eigene und gegnerische Kräfte ein.

Fadenkreuz: Zeichnet ein Fadenkreuz auf den Bildschirm

Automatisch gelenkte Panzerabwehr: Lenkt Anti-Tank Raketen von Geisterhand ins Ziel.

Uhranzeige: Gibt dem Spieler die Möglichkeit eine Uhr zu nutzen.

Beobachtungsperspektive: Erstellt eine Kamera hinter und über der Spielerfigur. Diese folgt dem Spieler. (sehr unrealistisch)

Leuchtspur: Zeigt eine Leuchtspur hinter einigen Projektilen. Damit wird eine Flugbahn sichtbar.

Super KI: Bestimmt ob die KI automatisch Angriffsaktionen ausführen soll.

Automatisches Zielen: Lenkt die Waffe automatisch auf ein Ziel. (Kinderkram)

Unbegrenzt Speichern: Gibt dem Spieler die Möglichkeit in Missionen unbegrenzt oft zu speichern.

Weitergehend besteht die Möglichkeit eine Fähigkeit für die feindlichen und freundlichen Truppen zu treffen. (Clanspieler spielen im Veteran Modus, ohne Außensicht und Fadenkreuz und haben die Truppen auf Maximum gesetzt)

Es können noch Untertitel und Funksprüche aktiviert oder abgeschaltet werden. (hierbei handelt es sich um die blauen Nachrichten welche bei Funksprüchen auftauchen)

Mit OK werden die Einstellungen übernommen, mit Abbrechen wird der Bildschirm nicht übernommen. Mit Voreinstellung wird die Vorgabe des Spiels geladen. Einige Einstellungen werden erst mit Neustart der Mission wirksam.

Rasenmäher

Hier ist eine kleine Hilfe beschrieben um sich dem Gras in Arma zu entledigen. So eine Art Rasenmäher. Viele leistungsarme Maschinen haben gerade im MP das Problem das sie bei großen Grasflächen lange Bildaufbauzeiten haben. Dabei bleibt der Spielspaß auf der Strecke.

Die Server Methode:

In der config vom Server den Wert für Gras auf 0 setzen. Damit ist das Gras ausgeschaltet. Humm Befehl geht nicht. Liegt an der beta. Sorry

Client Idee. Einfach in der Config der Inseln das Gras entfernen.

```
class GrassFlowers: GrassGeneral {
    model = "ca\plants\clutter_grass_flowers.p3d";
};

class GrassLong: GrassGeneral {
    model = "ca\plants\clutter_grass_long.p3d";
    affectedByWind = 0.600000;
    scaleMin = 0.600000;
    scaleMax = 1.100000;
};

class GrassSevenbeauty: GrassGeneral {
    model = "ca\plants\clutter_grass_sevenbaeuty.p3d";
    affectedByWind = 0.200000;
    scaleMin = 0.700000;
    scaleMax = 1.100000;
};

class GrassYellow: GrassGeneral {
    model = "ca\plants\clutter_grass_yellow.p3d";
    affectedByWind = 0.200000;
    scaleMin = 0.700000;
    scaleMax = 1.100000;
};

class GrassDesert: GrassGeneral {
    model = "ca\plants\clutter_grass_desert.p3d";
};
...

```

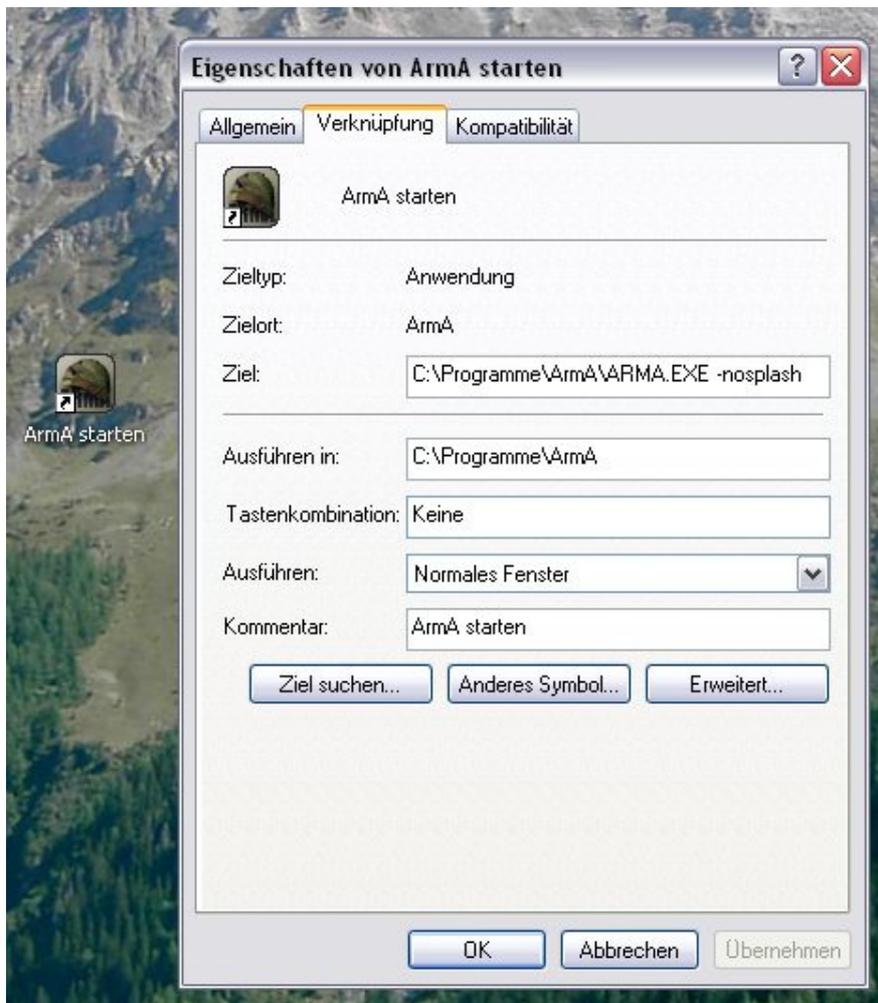
Einfach die entsprechenden Einträge entfernen. model = "";

Allgemeine Hilfe in ArmA

Armed Assault – Parameter

Wer Operation Flashpoint kennt und schon gespielt hat wird die Unmenge an Addons erlebt haben. Auch weiß der eine oder andere OFP Veteran, wie man die alte Dame weiterhin auf den Thron hieft. Das meiste trifft bei Armed Assault ebenfalls zu, jedoch gibt es technisch bedingt manches Kürzel nicht mehr, auch sind wiederum neue dazu gekommen. In diesem Tutorial wird eine anschauliche, beispielhafte Darstellung der Konfigurationen versucht aufzuzeigen.

Um diese Kürzel zu verwenden, muss man eine Verknüpfung anlegen. In dieser Verknüpfung kann man dann im Ziel die Kürzel einbinden.



Video-Optionen

Kürzel

- x= Numerisch, um Breite der Auflösung einzustellen
- y= Numerisch, um Höhe der Auflösung einzustellen
- window Anzeige erfolgt im Fenster, statt als Vollbild
- nosplash Deaktiviert den Standard-Ladebildschirm. Laut BIS bringt dieses Kürzel einen nur geringfügigen Performanceschub, da eigentlich während des Ladescreens die Spieldaten geladen werden und diese ja ebenfalls mit dem alternativen Start benötigt werden.
- nomap Addons werden nur geladen, wenn sie benötigt werden und verhindert memorybedingte Fehler beim Start des Spiels.
- benchmark Sollte eigentlich zum Benchen gedacht sein, ist aber nie fertig gestellt worden und funktioniert nicht.

Verschiedenes

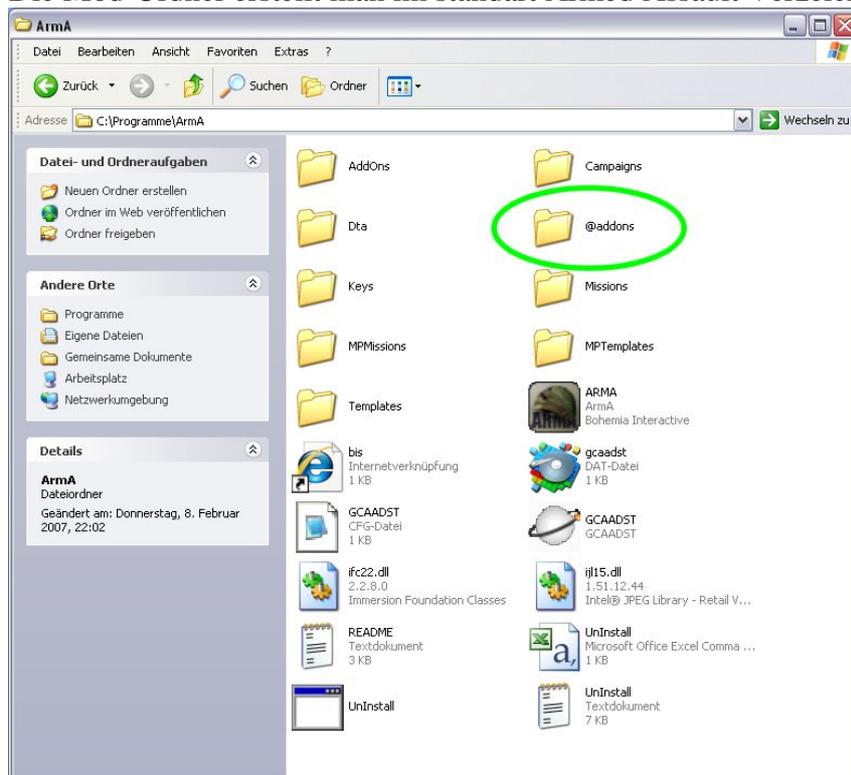
- maxmem= RAM-Speicherbegrenzung in MB
- init=Startet Scriptbefehle, wenn man im Menü ist, um zum Beispiel gleich mit dem Start von ArmA eine Mission zu starten:

```
„-init=playMission[, 'M04Saboteur.Sara']"
```
- world= Läd eine Insel vor wie zum Beispiel -world=Sara
- world=empty Sorgt für schnelleres Starten, da keine Insel und keine Menüanimationen geladen werden
- noland Startet ArmA ohne Insel (Für die Nutzung des Bulldozers)

Modifikationen

-mod= Erlaubt dem Nutzer die Addons in verschiedene Ordner aufzusplitten. Man unterteilt seine Addons in verschiedene Ordner. Der Vorteil, den man durch dieses separate Laden erreicht, zeichnet sich in der Performance aus, da nicht alle, sondern nur die benötigten Addons geladen werden müssen.

Die Mod-Ordner erstellt man im standart Armed Assault Verzeichnis.



Diese Ordner können beliebige Namen besitzen, sollten jedoch mit einem „@“ beginnen ein Beispiel: „@addons“ oder „@cti“ (Ohne „-“).

In diesen Ordner müsst ihr erneut einen weiteren Ordner erstellen, der in „addons“ umbenannt werden muss. Dort könnt ihr dann eure Addons hinein packen. Hier mal ein Pfad, der zu einem Addon führt:

C:\Programme\Bohemia Interactive\ArmA\@addons\addons\vilas_mod.pbo

Um einen Mod-Ordner mit ArmA zu starten, müsst ihr in der Zielangabe der Verknüpfung (siehe oben) folgendes angeben:

Wir wollen zum Beispiel, dass die Addons in dem Mod-Ordner „@addons“ geladen werden. Wir müssen also nun die Zeile „-mod=@addons“ eingeben (mit Leerzeichen vor dem „-“). Es besteht auch die Möglichkeit mehrere Ordner zu laden, man muss aber die Ordnernamen mit einem Semikolon trennen. Die Zeile sollte dann so aussehen: **-mod=@addons;@cti**





Vorteile bestehen darin, dass weniger Speicher benötigt wird, da nur ausgewählte Addons geladen werden, originale Daten/ sich überschneidende Addons voneinander getrennt werden und so bei Kompatibilitätsproblemen einfach der Ordner ausgewechselt bzw. gelöscht werden kann.

-cfg= Wählt eine Config, die man will. Wie die „arma.cfg“.
 -profiles Ist eine Startoption, die es ermöglicht alternative Ordnerangaben des Nutzerprofiles, sowie Missionen und .cfg Dateien zu laden
 Der Windows Benutzeraccount würde die nötigen Rechte benötigen, um den gewählten Ordner zu erstellen. Beispiel:

C:\Program Files\Bohemia Interactive\ArmA\ArmA.exe" -profiles=C:\ArmA\Profiles

Einfügen von Custom Faces

Custom Faces sind eine einfache Möglichkeit seinen Missionen und seiner Spielfigur in Armed Assault etwas Persönlichkeit einzuhauchen.



Nun tritt häufig die Frage auf: Wie bekomme ich das Custom Face in mein Spiel?

Hier ist die Antwort:

Als erstes besorgt ihr euch ein Custom Face (in Form einer Bilddatei), zu finden auf diversen Community Seiten.

Die Datei speichert ihr unter dem Namen „face.jpg“ in eurem „ArmA Other Profiles“ Ordner, der Standard Pfad ist :

„C:\Dokumente und Einstellungen\Benutzername\Eigene Dateien\ArmA Other Profiles\Nickname“.

!Achtung: Die Datei darf nicht größer als 96kb sein!

Nun startet ihr Armed Assault, klickt oben links auf euer Profil und wählt im folgenden Bildschirm „Bearbeiten“ aus. Oben rechts im Auswahlfenster für das Gesicht eurer Spielfigur klickt ihr auf „Individuelles Gesicht“ und schon müsste euer ausgewähltes Custom Face links auf dem Kopf zu sehen sein. Das oben zu sehende Gesicht sieht im Spiel so aus:

Einfügen von Custom Sounds

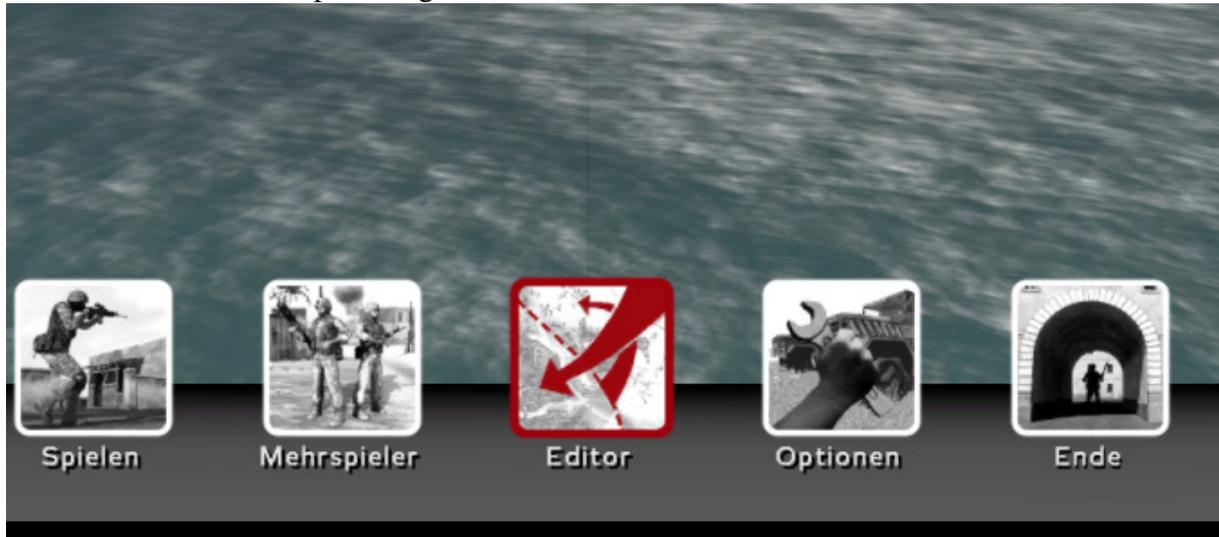
In den Foren findet man des öfter die Frage nach Soundstücke welche im MP abspielt werden. Das System ist relativ einfach anzuwenden. Hierzu wird im Profileordner

„C:\Dokumente und Einstellungen\Benutzername\Eigene Dateien\ArmA Other Profiles\Sound“.

des Spielers einfach ein Ordner mit dem Namen Sounds angelegt. In diesen werden die Soundfile geladen. Bitte nehmt davon Abstand diese zu groß zu machen, da jeder Spieler diese auf einem Server laden muss. Das gilt auch für euer Gesicht. (Viele Server haben Begrenzungen was die Größe des Ordners angeht.)

Starten des Editors

Der Editor wird im Hauptmenü geöffnet. Der nächste Schritt ist dann die Insel auszuwählen.

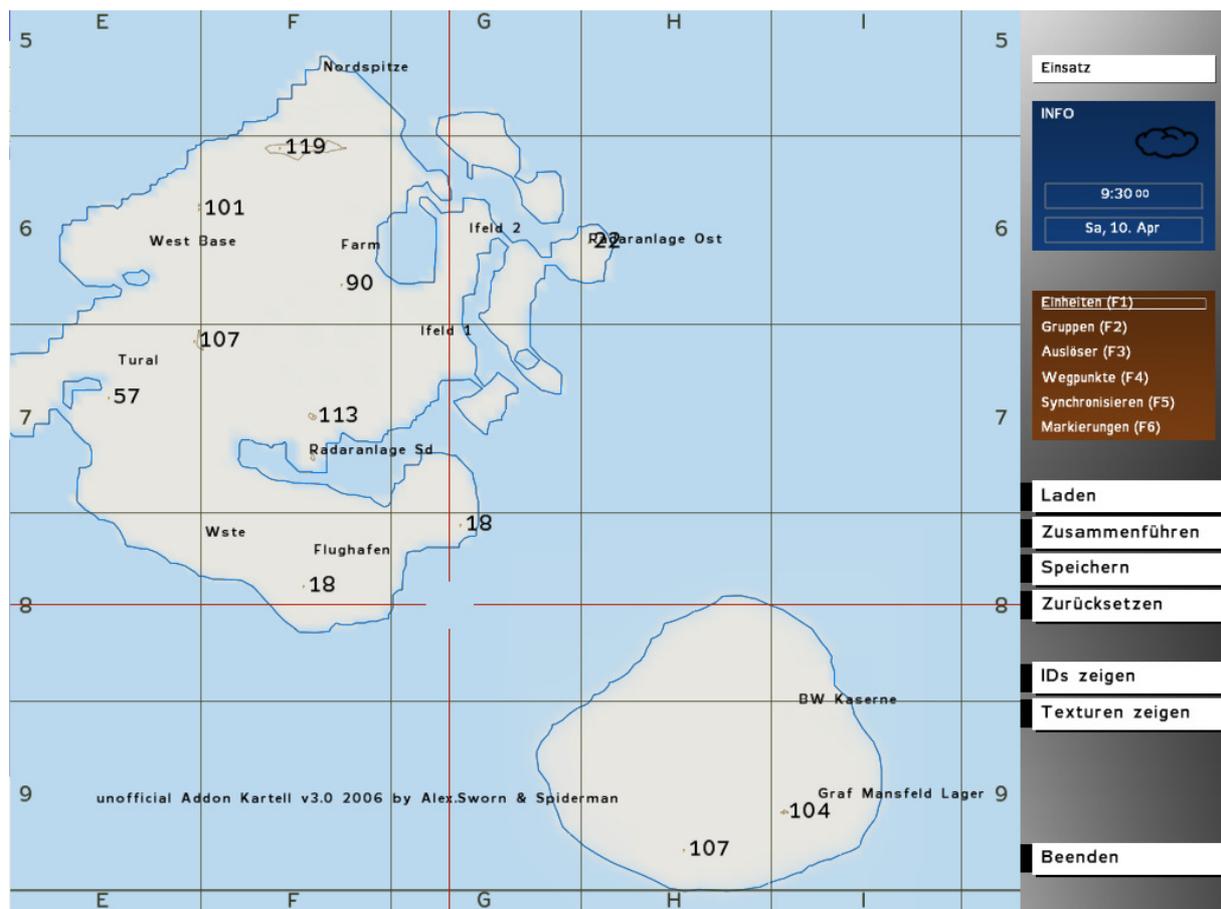


Unter Editing wird der Editor geöffnet. Zuerst hat der Spieler hier die Möglichkeit die Insel zu wählen auf der er seine Mission bearbeiten möchte.



In diesem Beispiel ist die OFW - Insel v1.0 ausgewählt. Nach dem Laden der Insel kann diese dann editiert werden. (Hier sind alle BI Inseln sowie alle mitgeladenen freien Inseln aufgeführt.). Auswahl erfolgt über Doppelklick oder durch Auswahl und Bestätigung mit OK.

Der Editor



Einstellungen im Editor.

Unter dem Menü Button „Einsatz“ werden vier Unterpunkte angezeigt:

„**Einsatz**“: Diese Einstellung dient dazu auf der jeweilig geladenen Karte die ein Spiel zu erstellen und Handlungsabläufe mittels Auslöser zu gestalten (Editieren).

„**Einleitung**“: Ist diese Funktion aktiviert, ist es möglich einen Vorspann für eine Mission zu gestalten. Meistens wird hier mit einem Kamerascript eine kurze Vorgeschichte erklärt.

Achtung: diese Option ist nur für Singleplayer Einsätze gedacht. Soll die Mission Multiplayer fähig sein, so muss die Einleitung direkt in der Mission (also unter der Einstellung „Einsatz“ erstellt werden.

„**Abspann – verloren**“: Diese Option ist ebenfalls nur für Singleplayer Einsätze gedacht. Per Auslöser ist diese Option anwählbar. Näheres siehe Kapitel: Auslöser

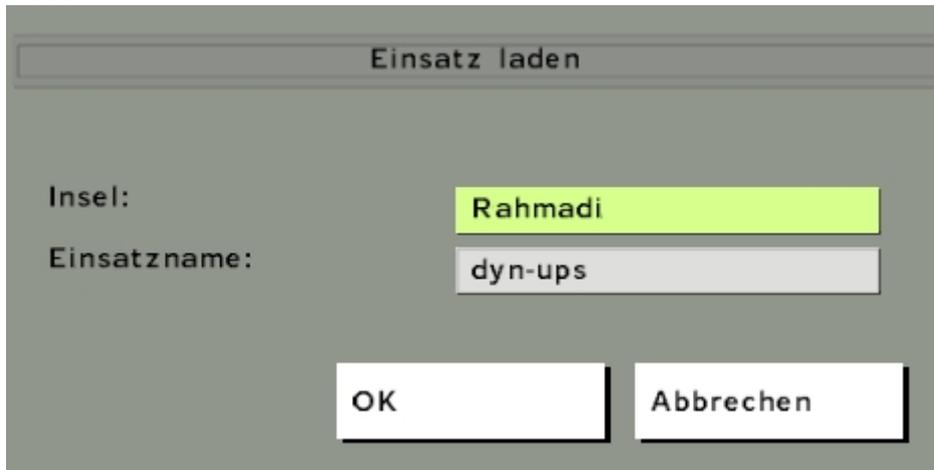
„**Abspann – gewonnen**“: das gleiche wie Punkt 3.

Der Infoschirm:

Hier kann das Wetter, die Uhrzeit, der Name und eine kurze Beschreibung der Mission verfasst werden. Die Seite auf der der Widerstand kämpfen soll wird hier ebenfalls bestimmt. Darunter erfolgt die Auswahl Einheit F1, Gruppe F2, Auslöser F3, Wegmarke F4, Synchronisieren F5, Markierungen F6.

Es folgen die Menübefehle:

Laden: lädt einen erstellten Missionsordner



Zusammenführen: Mehrere Missionsordner können für den Singleplayer zusammen gefügt werden

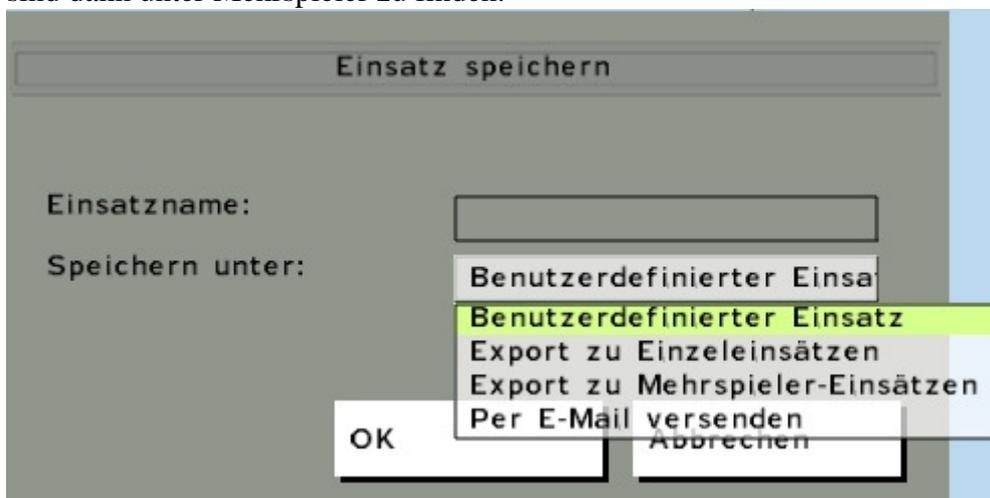
Speichern: Speichert den editierten Stand in der mission.sqm

Es besteht die Möglichkeit diese als „benutzerdefinierte Mission“, „Einzelspieler“, „Multiplayer“ oder als „E-mail verschicken“ zu speichern.

Die benutzerdefinierten Missionen werden im Benutzerordner Missionen gespeichert und sind nicht zu einer *.pbo gepackt.

Die Einzeleinsätze werden im Arma Ordner Missionen als *.pbo abgelegt. Diese sind dann unter Einsätze zu finden..

Die Mehrspieler Einsätze werden im Arma Ordner MPMissionen als *.pbo abgelegt. Diese sind dann unter Mehrspieler zu finden.



Zurücksetzen: Löscht alle editierten Einheiten auf der Karte

ID zeigen: Jeder Gegenstand auf der Karte (Häuser, Objekte) hat eine Nummer. Mit dieser Option lassen sich die Objektnummern anzeigen

Texturen zeigen: zeigt die Texturen auf. zB: Wüstenfläche, Grünefläche...

Vorschau: Nur möglich wenn eine als Spieler definierte Einheit im Editor platziert wurde (per Doppelklick auf die Karte). Startet das „Preview“ im Editor. Mit der „Esc“ Taste gelangt man wieder in die Kartenansicht

Weiter: Führt den Spieler zurück zur editierbaren Kartenansicht

Beenden: Beendet den Editor

Die Tasten F1 – F6

Mit den F-Tasten ist ein schneller und einfacher Gebrauch des Editors möglich. Man spart sich den Klick auf die Editor Menüleiste, was bei längerem Umgang mit dem Editor viel Zeit spart.

Hier eine kleine Übersicht über die Funktion der F-Tasten:

F1 – Einheit Hinzufügen

F2 – Gruppe Hinzufügen

F3 – Auslöser einfügen

F4 – Wegpunkt einfügen

F5 – Synchronisieren

F6 – Marker einfügen

Auf den nächsten Seiten werden die F-Tasten und deren Funktion genauer erklärt und erläutert.

Einheit Hinzufügen F1

Um eine Einheit zu erstellen drückt man die F1 Taste oder man klickt auf den Button „Units“ rechts im Editor Menü. Um nun die Einheit, das Fahrzeug oder das Objekt einzufügen, klickt man hierzu doppelt (Doppelklick mit der linken Maustaste) auf die Stelle der Editor Karte, an der das Objekt, das Fahrzeug oder die Einheit erstellt werden soll:

Einheit hinzufügen

Seite: Westen Dienstgrad: Private

Klasse: Männer Einheit: Schütze

Kontrolle: Nicht spielbar Speziell: In Formation

Info-Alter: Unbekannt Name:

Fahrzeug Zugang: Voreinstellung Fähigkeiten:

Initialisierung:

Beschreibung:

Gesundheit / Par

Treibstoff

Munition

Anwesenheit (Wahrscheinlichkeit):

Anwesenheit (Bedingung): true

Radius der Platzierung: 0

Azimut: 0

OK Abbrechen

Hier werden die Informationen der zu erstellenden Einheit bestimmt:

Seite: Westen, Osten, Widerstand, Zivilist, Spiellogik, Leer Funktion:

Hier kann man die Seite der einzufügenden Einheit wählen. Um Spiellogik und Leer selektieren zu können muss man schon zuvor eine Einheit eingefügt haben die das Attribut „Spieler“ hat (siehe „Kontrolle“)

Klasse: Männer, Luft, Gepanzert, Auto, Schiff, Statisch, Unterstützung, Objekte, Munition, Mine, Sounds

Hier wählt man für die jeweilige Seite die Art der Einheit aus. Es wird bestimmt welcher Hauptgruppe eine Einheit angehört.

Kontrolle: Spieler, Spielbar, nicht Spielbar, Spieler Spielbar als Kommandant, Schütze, Fahrer
Hier wird bestimmt ob die Einheit Spielbar (Multiplayer Definition), oder die Spieler Einheit ist. Ist bei Klasse ein Panzer gewählt, besteht noch die Möglichkeit den Spieler als Comander Schütze oder Driver zu bestimmen oder welche Rolle im Multiplayer Spiel freigegeben ist – Spielbar als...

Info-Alder: Tatsächlich, 5Minuten, 10Minuten, 15Minuten, 30Minuten, 1Stunde, 2Stunden, Unbekannt

Bestimmt die Info die der Spieler über diese Feindeinheit hat. Cadet modus

Fahrzeug Zugang: Offen, Voreinstellung, Abgeschlossen

Hier wird festgelegt ab der Spieler in das Fahrzeug einsteigen kann. Dies kann natürlich über ein Skript oder Auslöser geändert werden. Name lock true für verschlossen oder Name lock false für offen.

Dienstgrad: Private, Corporal, Sergeant, Lieutenant, Captain, Major, Colonel

Bestimmt die Rangordnung im Spiel, regelt die Befehlskette einer Gruppe

Einheit: Je nach Klasse sind hier die Einheiten aufgeführt.

Hier wird der Typ der Klasse ausgewählt. M1A2 oder SF oder Auto gelb...

Speziell: Keine, Im Laderaum, Fliegen, In Formation

(Bestimmt wo die Einheit erstellt wird.)

„Keine“ – Einheit steht auf dem ihr im Editor gegebenen Position und begibt sich dann in Formation. Dies kann verhindert werden wenn in der Initialisierung dostop this steht.

„In Formation“ - lässt die Einheit in der vom Führer der Gruppe bestimmten Position entstehen (Grundeinstellung: Formation Keil).

„Im Laderaum“ - sitzt die Einheit wenn vorhanden in den Cargo eines Transporters.

„Fliegen“ - setzt die Einheit mit laufendem Motor in eine Höhe von 50m.

(Dies gilt für Luftfahrzeuge und Fallschirme)

Name: Hier wird der globale Name der Einheit bestimmt. Dies ist wichtig wenn die Einheit später durch einen Auslöser, Skript gezieht angesprochen werden soll.

(Dieser kann nur einmal vergeben werden.)

Fähigkeit: Der Regler stellt die Fähigkeiten der AI ein die diese Einheit hat.

Initialisierung: hier können der Einheit Skripte etc mitgegeben werden.

(Hier eingetragene Befehle werden nach Missionsstart aufgerufen. Es können zum Beispiel Skripte oder Verhaltensaufforderungen sein.)

Beschreibung: Hier kann ein Text eingefügt werden. Dieser ist im „Switch Unit“ Bildschirm zu lesen.

Gesundheit / Par: Schieberegler um die Gesundheit festzulegen. Dies kann auch durch einen Befehl (Name setdammage Wert) ausgeführt werden.

Treibstoff: Schieberegler um die Treibstoffmenge zu bestimmen. Dies kann auch durch einen Befehl (Name setfuel Wert) ausgeführt werden.

Munition: Schieberegler um die Munitionsmenge zu bestimmen. Dies kann auch durch einen Befehl (Name setAmmoCargo Wert) ausgeführt werden.

Anwesenheit (Wahrscheinlichkeit): Schieberegler um die Wahrscheinlichkeit der Einheit zu bestimmen.

Radius der Platzierung: Hier kann ein Radius eingegeben werden. Dieser bestimmt die Entfernung in der die Einheit erstellt werden kann. Dazu kommt noch die Kompassrose an der die Ausrichtung der Einheit bestimmt werden kann. Hierzu wird einfach ein Wert 0 bis 360 eingegeben oder mit der Maus ein Winkel in der Rose angeklickt.

Nach erstellen der gewünschten Einheit mit OK oder Enter bestätigen.

Gruppe Hinzufügen F2

Um eine Gruppe zu erstellen kann durch drücken der Taste F2 oder durch Auswahl des Button Gruppe und folgendem Doppelmausklick auf die Karte dieser Bildschirm geöffnet werden.



Dies ermöglicht eine vorbestimmte Gruppe zu erstellen. Die Gruppe kann natürlich weiter bearbeitet werden. Die Gruppenmitglieder werden im Modus „In Formation“ erstellt. Die Option dient dem schnellen erstellen von Missionen. Um eine Einheit einer Gruppe hinzuzufügen oder aus einer Gruppe heraus zu nehmen wird nach Auswahl der Gruppe (mittels linker Mausklick und Markierung der Einheiten) F2 aktiviert. Nun ist es möglich eine Einheit mit linker Maustaste anzuwählen. Wird die entstehende blaue Linie ins nichts geführt, ist diese Einheit nicht mehr im Gruppenverband. Um eine Einheit zuzufügen – links klick und blaue Linie auf ausgewählte Gruppe ziehen. Die neu eingefügte Einheit nimmt automatisch durch ihren eingestellten Dienstgrad ihre Funktion in der Gruppe war.

Seite: Westen, Osten, Widerstand, Zivilist

Typ: Bestimmung der vorbereiteten Gruppeart

Name: Bestimmt den Typ der Gruppe.

Kompassrose: Hier kann die Ausrichtung der Einheit bestimmt werden kann. Hierzu wird einfach ein Wert 0 bis 360 eingegeben oder mit der Maus ein Winkel in der Rose angeklickt. Nach erstellen der gewünschten Gruppe mit OK oder Enter bestätigen.

Auslöser hinzufügen F3

Um einen Auslöser zu erstellen wird im Editor durch drücken der Taste F3, oder durch Auswahl des Button Auslöser und folgendem doppel Mausklick auf der Karte dieser Bildschirm geöffnet.

Auslöser hinzufügen

Achse A Achse B

Winkel

Aktivierung

Countdown Timeout Min.: Max.: Mid.:

Typ

Name:

Bedingung

Bei Aktivierung

Bei Deaktivierung

Auslöser dienen in erster Hinsicht zum bestimmen von Ereignissen und zur Abfrage von Zuständen. Mit ihnen können eine ganze Reihe von Bedingungen abgefragt werden. Sie sind das Herzstück einer Mission. Sie sind eine Art ON Off Schalter für Missionsereignisse.

Achse A: Hier wird die horizontale Größe des Auslösebereiches festgelegt.

Achse B: Hier wird die vertikale Größe des Auslösebereiches festgelegt.

Winkel: Dem bestimmten Bereich wird ein Winkel bestimmt.

Ellipse: Durch Auswahl wird der Bereich als Kreis bestimmt.

Rechteck: Durch Auswahl wird der Bereich als Rechteck bestimmt.

Aktivierung: Hier wird bestimmt wer den Auslöser aktiviert. Das kann über eine Seite geregelt werden oder über ein Funk. Weitergehend besteht die Möglichkeit durch drücken der Taste F2 oder durch Auswahl des Buttons „Gruppe“ und halten der linken Maustaste eine blaue Linie von einer Einheit auf den Auslöser zu ziehen. Dann wird der Auslöser nur von dieser Einheit an oder ausgeschaltet. Es besteht dann die Möglichkeit zu wählen ob nur diese Einheit oder jede Einheit der Gruppe für die Auslösung in Frage kommt.

Einmal: Nur einmaliges auslösen möglich.

Mehrfach: mehrmaliges auslösen möglich.

Vorhanden: Auslöser wird aktiviert wenn eingestellte Seite im Bereich des Auslöserbereichs ist.

Nicht Vorhanden: Auslöser wird aktiviert wenn eingestellte Seite nicht im Auslöserbereich ist.

Von West entdeckt: Löst aus wenn eine Einheit der bestimmten Seite von Westeinheiten entdeckt wird.

Von Osten entdeckt: Löst aus wenn eine Einheit der bestimmten Seite von Osteinheiten entdeckt wird.

Von Zivilisten entdeckt: Löst aus wenn eine Einheit der bestimmten Seite von Zivilisten entdeckt wird.

Countdown/Timeout: Hier werden Verzögerung, Zeitpunkt eingestellt. Der Auslöser kann so dynamisch angepasst werden ob auf ein Ereignis sofortige oder verzögerte Auslösung erfolgt.

Min.: Minimale Verzögerung zwischen dem Auslösen und des Ausführens.

Max.: Maximale Verzögerung zwischen dem Auslösen und des Ausführens.

Mid: Ein Richtwert der das bestimmen des Zufallswerts zwischen Min. und Max. beeinflusst

Typ: Hier wird die Art des Auslösers bestimmt. Dies ist für Ende einer Mission oder als Schalter für Wegmarken geeignet.

Name: Hier wird dem Auslöser ein Name gegeben. Dies ist wichtig wenn dieser später durch ein Skript oder Auslöser angesteuert, verschoben etc. werden soll.

Text: hier kann eine kurze Beschreibung des Auslösers erfolgen. Es ist auch die Anzeige die erscheint wenn man als Typ eine Radio Aktivierung wie Radio: Alpha ausgewählt hat.

Bedingung: Hier können unabhängig der allgemeinen Einstellungen Bedingungen für die Aktivierung bestimmt werden. Es besteht die Möglichkeit viele zu kombinieren oder mit den allgemeinen Bedingungen zu kombinieren. (and, or, not, siehe Skripting Befehle).

Bei Aktivierung: Hier können Skripte, Aktionen und Bedingungen geschaltet und gestartet werden.

Bei Deaktivierung: Hier können Skripte, Aktionen und Bedingungen geschaltet und gestartet werden.

Nach erstellen des gewünschten Auslösers mit OK oder Enter bestätigen. Weitergehend besteht die Möglichkeit Effekte einzubinden.

Effekte im Auslöser.

Effekte bearbeiten	
	true
Anonym:	Kein Sound
Stimme:	Kein Sound
Umgebung:	Kein Sound
Auslöser:	Kein Sound
Spur:	Keine Musik
Typ:	KEINEN
Effekt:	EINFACH
<input type="button" value="OK"/> <input type="button" value="Abbrechen"/>	

true: Hier können noch Bedingungen für das ausführen eines Effektes bestimmt werden. Es besteht die Möglichkeit hier Sound, Musik, Stimmen oder Texteinblendungen einfach zu starten.

Anonym: Hier kann ein Sound bestimmt werden welcher von einer anonymen Person kommt.

Stimme: Hier kann ein Sound bestimmt werden welcher aus einer Richtung kommt

Umgebung: Hier werden Umgebungsgeräusche ausgewählt. Grille, Hund, Vögel.

Auslöser: Hier sind 3d Sounds aufgeführt. Gewehrfeuer, Haus Einsturz, Granaten.

Spur: Hier kann ein Musikstück eingespielt werden. Eigene Musik wird hier auch gefunden.

Typ: Objekt (Hier können Bildeinblendungen bestimmt werden)

Ressource: Hier können Bildeinblendungen oder Kameraeffekte bestimmt werden.

Text: es öffnet sich ein Textfeld in dem der Spieler Texteinblendungen machen kann

Effekte: Bestimmt den Übergangseffekt zu einer Text oder Bild Einblendung

Wegpunkte hinzufügen F4

Um einen Wegpunkt zu erstellen wird im Editor durch drücken der Taste F4 oder durch Auswahl des Buttons „Wegpunkt“ und folgendem Doppelmausklick auf die Karte dieser Bildschirm geöffnet.

Wegpunkt hinzufügen

Typ auswählen:	BEWEGEN
Wegpunktfolge:	0:
Beschreibung:	
Kampfmodus:	Keine Veränderung
Formation:	Keine Veränderung
Geschw.:	Keine Änderung
Verhalten:	Keine Änderung
Platzierungsradius:	0

Timeout	Mindest	0	Höchste	0	Mittelw	0
Bedingung:	true					
Bei Aktivierung:						
Skript:						

Nie anzeigen

Kadettmodus anzeige

Immer anzeigen

Effekte

OK

Abbrechen

Hier wird für eine Gruppe oder Einheit eine Wegmarke erstellt. Die Einheit arbeitet die erstellten Wegpunkte der Reihennfolge nach ab. Durch folgende Einstellungen lässt sich das Verhalten der Einheit von Wegpunkt zu Wegpunkt beeinflussen:

Typ auswählen

Bewegen: Einheit bewegt sich zum nächsten Wegpunkt

Zerstören: Einheit attackiert alle Feindeinheiten in seiner Umgebung

Einsteigen: Einheit steigt in Fahrzeug ein

Suchen und Zerstören: Einheit attackiert alle Feindeinheiten in seiner Umgebung und sucht

die Umgebung ab

Anschliessen: Einheit tritt Gruppe bei

Anschliessen und Führen: Einheit tritt Gruppe bei. Gruppe folgt Einheit auf dessen Wegpunkten

Aussteigen: Einheit steigt aus Fahrzeug aus

Wiederholen: Einheit beginnt die Wegpunkte erneut abzuschreiten (patrouilliert)

Laden: Siehe synchronisieren

Entladen: bis auf Anführer steigt Gruppe aus Fahrzeug aus

Transport Entladen: Alle Einheiten der Gruppe steigen aus

Halten: Führt die Wegpunktkette hinter diesem Punkt nicht weiter

Aufklären: rückt vorsichtig zum nächsten Wegpunkt vor

Bewachen: bleibt stationär und attackiert sofort Feindeinheiten

Spreschen: dient dem Auslösen von Lip Dateien

Gescriptet: hier kann eine Script Syntax gestartet werden

Unterstützen: synchronisiert mit einer anderen Einheit, lässt beide Einheiten gleiches Feindwissen haben

In nächstes Einsteigen: Einheit steigt in nächstes Fahrzeug ein

Entlassen: Einheit scheidet aus Gruppe aus

Wegpunktfolge: Zeigt die Nummer des Wegpunktes an

Beschreibung: Notizen für den Wegpunkt möglich

Kampfmodus: Keine Veränderung, Nie schießen, Nicht schießen, nicht schießen Angriff nach eigenem Ermessen, Feuer eröffnen, Feuer eröffnen Angriff nach eigenem ermessen (blue, green, white, yellow, red)

Formation: keine Veränderung, Kolonne, Gestaffelte Kollonne, Keil, Staffel links, Staffel rechts, V-Förmig, Reihe, Delta, Kolonne kompakt

Geschwindigkeit: Begrenzt, Normal, Voll

Verhalten: Achtlos, Sicher, Wachsam, Kampf, Verdeckt

(Careless, Safe, Aware, Combat, Stealth)

Platzierungsradius: Gibt ein kreisförmiges Gebiet an indem die Einheit erstellt wird, oder sich bewegt

Timeout: Einstellen einer Zeitspanne welche die Einheit am Wegpunkt verweilen soll. Es besteht die Möglichkeit einen minimalen und einen maximalen, sowie einen Richtwert zu bestimmen. Die Einheit verweilt dann eine Zeit (zwischen min und max) am Wegpunkt

Bedingung: Hier können Bedingungen eingefügt werden die erfüllt werden müssen bevor dieser passiert werden kann. z.B. Alarm; Erst wenn vorher durch ein Skript oder Auslöser Alarm=true gesetzt wurde werden sämtliche nachfolgende Wegpunkte abgeschritten

Bei Aktivierung: Hier kann eine Bedingung auf wahr oder unwahr gesetzt werden

Skript: Hier kann ein Skriptaufwurf gestartet werden

Ob der Wegpunkt im Spiel gezeigt werden soll lässt sich wie folgt einstellen: (Abhängig vom gewählten Spielschwierigkeit – Cadet – Veteran)

-Nie anzeigen

-Kadettmodus anzeigen

-Immer anzeigen

Effekte:

Zusätzlich zu den bereits beschriebenen Funktionen lässt sich bei Erreichen eines Wegpunktes ein Effekt einblenden. So kann bei Erreichen eines Wegpunkts ein Sound abgespielt werden oder eine Textmitteilung eingeblendet werden.

Hier können dieselben Effekte wie sie auch im im Kapitel Auslöser beschrieben sind eingebracht werden.

Synchronisieren F5

Es besteht die Möglichkeit Wegpunkte zu synchronisieren. Hierzu wird durch Tastendruck F5 oder durch Auswahl des Buttons „Synchronisieren“ und halten der linken Maustaste eine blaue Linie von einem Wegpunkt zu einem andern einer anderen Gruppe gezogen. Das sorgt dafür, dass die Gruppen erst weitergehen wenn beide den Wegpunkt erreicht haben. In diesem Beispiel sollen zwei Gruppen von Westen und Osten gleichzeitig in ein Dorf kommen.



Wie wir sehen, haben beide Gruppen verschiedene Strecken. Es ist also nicht möglich, dass diese zeitgleich eintreffen. Um das zu erreichen, wurden zwei Wegpunkte synchronisiert (blaue Linie). Die obere Gruppe wird daher vor dem Dorf warten, bis die untere Gruppe das Dorf umlaufen hat und bereit ist. Das kann natürlich auch mit mehreren Gruppen gemacht werden.

Weitergehend hat der Spieler die Möglichkeit Wegpunkte miteinander zu verknüpfen.

Einsteigen und Laden: Wenn in diesem Fall ein LKW den Punkt **LADEN** erreicht, würde er warten bis die Gruppe den Punkt **EINSTEIGEN** erreicht, diese dann aufnehmen und weiter fahren. Damit wird es möglich einen LKW zu einem Dorf fahren zu lassen, eine Gruppe dort abzuladen und in einem Bogen um das Dorf zu fahren. Die Gruppe könnte nach dem aussteigen das Dorf sichern und auf der anderen Seite wieder einzusteigen. Damit würde der LKW nicht in Gefahr kommen.

„**Anschliessen und Führen**“ und „**Anschliessen**“: Hierbei würde sich Gruppe A mit Gruppe B verbinden. Dabei sollte jedoch darauf geachtet werden, dass die Gruppe mit dem Wegpunkt „**Anschliessen und Führen**“ die Ranghöhere ist.

Markierung hinzufügen F6

Nach drücken der F6 Taste kann man eine Markierung einfügen welche im Missionsbriefing oder im Spiel auf der Karte angezeigt wird. Beim Einfügen muss man darauf achten einen „sinnvollen“ Namen für den Marker zu wählen, sodass man später beispielsweise einen Link im Missionsbriefing erstellen kann. Beim klicken auf den „Link“ im Briefing wird die Kartenansicht auf den verlinkten Marker zentriert. Zudem kann man mit Markern eine taktische Besprechung des Einsatzes visuell verdeutlichen.

Hier werden die Einstellungen eines Markers getroffen.

Name: Hier wird dem Marker ein Name gegeben. Dies ist wichtig wenn dieser später durch ein Skript oder Auslöser angesteuert, verschoben etc. werden soll, oder wie oben beschrieben im Briefing verlinkt wird.

Symbol: Es besteht nicht nur die Möglichkeit ein vorbestimmtes Zeichen zu verwenden. Es gibt auch die Möglichkeit einen eigenen Bereich zu erstellen dies kann entweder als Rechteck oder Ellipse erfolgen.

Farbe: Hier wird die Farbe des Markers gewählt.

Symbol: Hier wird der Typ des Markers (das Symbol) gewählt.

Achse A: Hier wird die Größe des Markers festgelegt.

Achse B: Hier wird die Größe des Markers festgelegt.

Winkel: Um einen Marker quer zu stellen besteht hier die Möglichkeit einen Winkel einzurichten.

Text: Hier kann eine kurze Beschreibung des Markers erfolgen der dann neben dem Marker auf der Karte erscheint.

Es gibt auf Community Seiten viele Addon Marker die das Markersortiment erweitern koennen, wobei jeder Spieler der Mission dieses Marker Addon braucht, da er sonst die Mission nicht spielen kann.

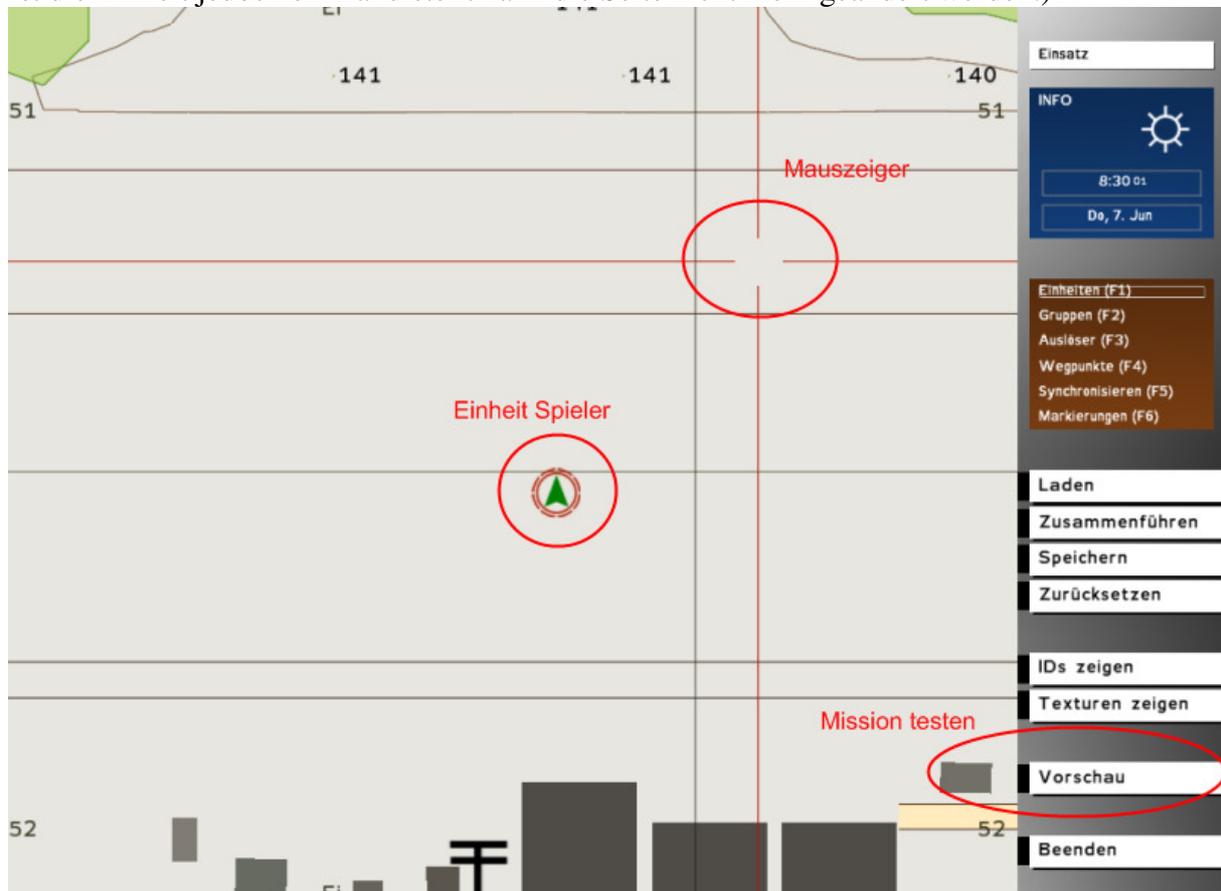
Editorbeispiele

Das kleine Mission ein mal eins.

Um den Editor kennen zu lernen sind hier einfache Beispiele aufgeführt mit denen der Umgang mit dem Editor erklärt wird.

Einheit erstellen

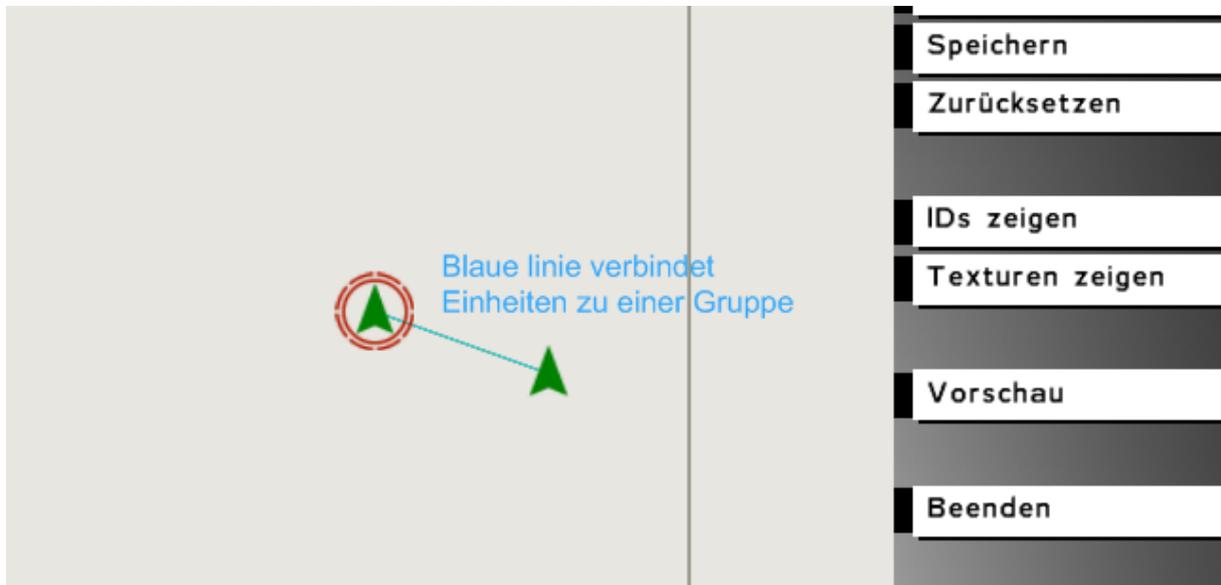
Öffnet den Mission Editor und wählt eine Insel. Im Editor wählt ihr dann durch drücken der F1 Taste oder durch auswählen per Mausclick die Option Einheit aus. Danach bringt den Mauszeiger auf die gewählte Position auf der Karte. Durch doppel Klick mit der linken Maustaste öffnet ihr jetzt der Bildschirm Einheit hinzufügen. (siehe F1 Einheit Hinzufügen) wählt da eure Einheit aus und bestätigt mit Enter oder durch drücken des OK Button. Jetzt ist eine Einheit auf der Karte platziert. Sobald eine Spielereinheit erstellt ist erweitert sich der Editor um die Option Vorschau. Mit Vorschau wird das Spiel gestartet und ihr könnt eure erste kleine Mission betrachten. (Erstellte Einheiten können in Typ Klasse geändert werden. Ist die Einheit jedoch einmal erstellt kann die Seite nicht mehr geändert werden.)



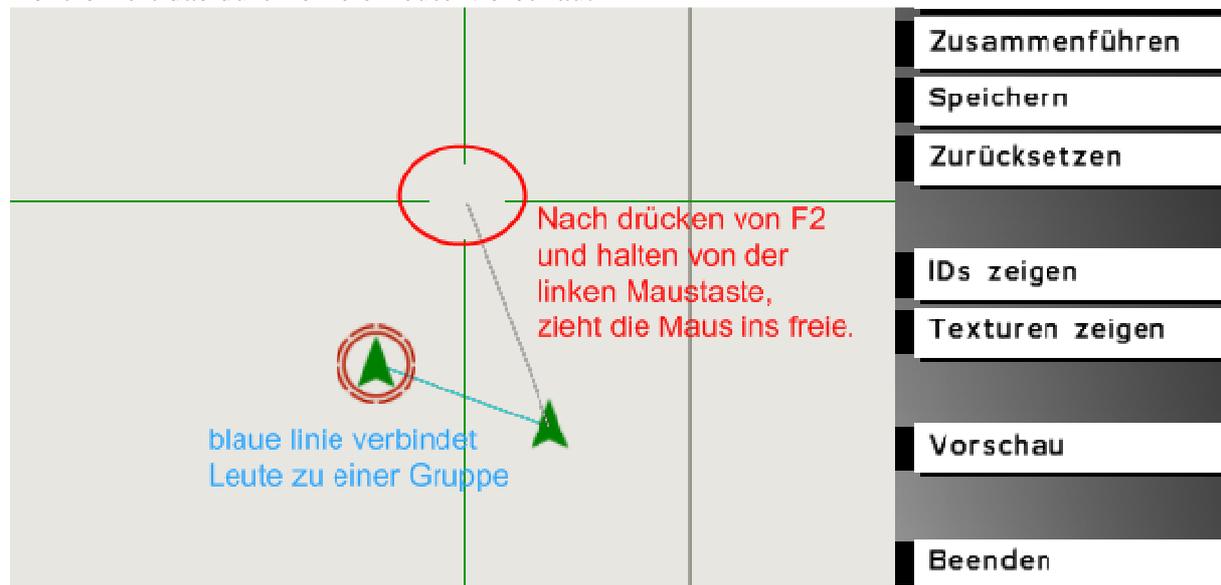
Durch drücken der Vorschau Taste kann das Ganze jetzt betrachtet werden.

Gruppenerstellen

Nun erstellen wir eine zweite Einheit neben der Spielereinheit. (Ändert nicht den Rang) Wenn die Einheit erstellt wurde sollte eine blaue Linie von der Spielereinheit zur zweiten Einheit laufen. (Die blaue Linie zeigt an, dass die beiden Einheiten zusammen eine Gruppe bilden.) Startet erneut über Vorschau die Mission. Ihr seid nun der Gruppenführer und könnt die Einheit über das Befehlsmenü mit Ordnern versehen.



Um die Einheit aus der Gruppe zu entfernen wählt die Option Gruppe. Dies geht durch F2 oder durch Auswahl von Gruppe. Dann zieht durch gedrückt halten der linken Maustaste eine blaue Linie von der Spielereinheit ins leere. Jetzt sind die beiden Einheiten wieder getrennt. Kontrolliert das durch eine erneute Vorschau.



Kennzeichnungen der Einheiten

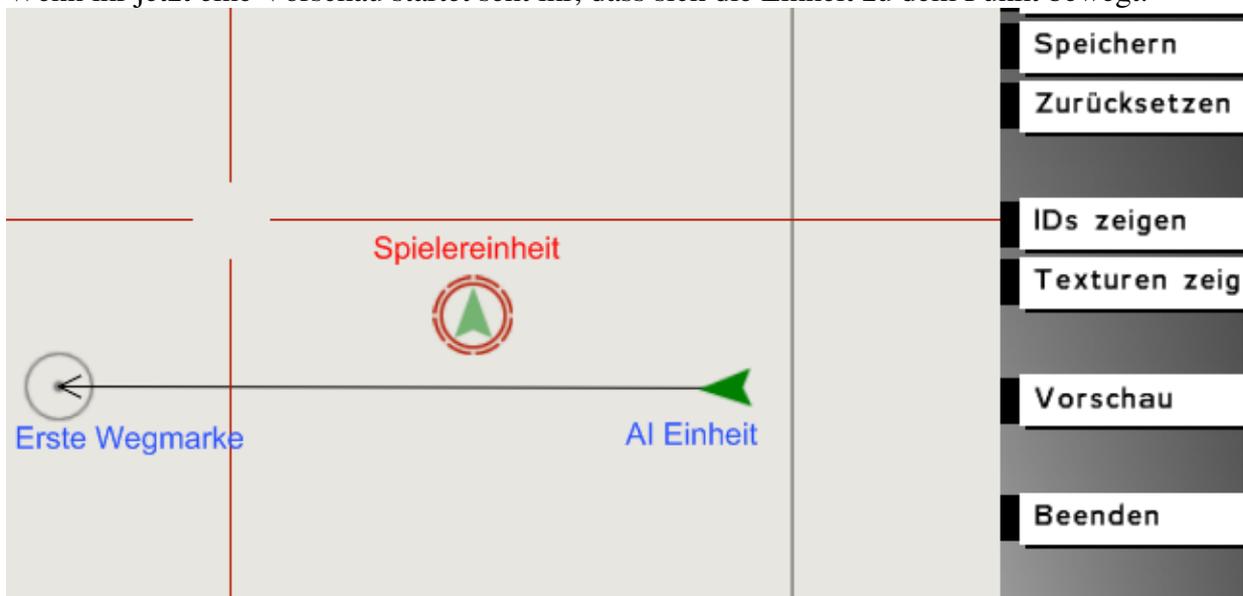
Im Editor können drei Einheiten unterschieden werden. Die Spielereinheit mit einer roten Umrandung, die Spielbare Einheit lila Umrandung und die AI Einheit ohne Umrandung.



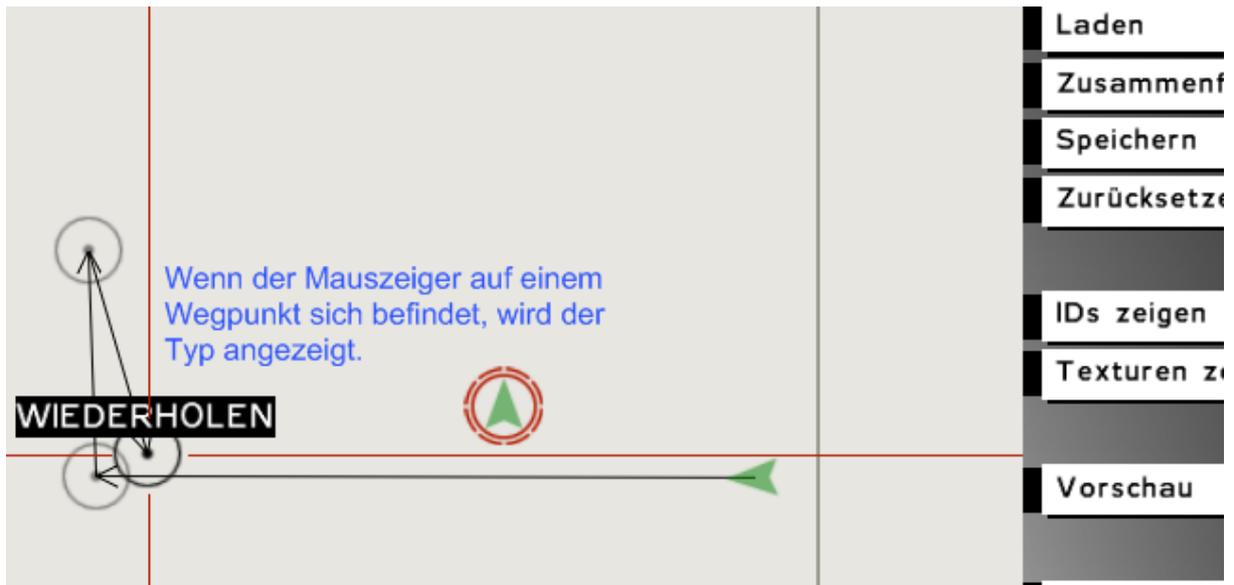
Wegpunkt

Jetzt wählt den Punkt Wegmarke F4 aus. Klickt einmal auf die AI Einheit neben euch und danach sucht eine Stelle auf der Karte. Erstellt da durch Doppelklick mit der linken Maustaste eine Wegmarke. (siehe Punkt F4 Wegmarke). Nicht die Eigenschaften der Wegmarke verändern.

Wenn ihr jetzt eine Vorschau startet seht ihr, dass sich die Einheit zu dem Punkt bewegt.



Nun wollen wir eine Einheit zwischen zwei Wegmarken patrouillieren lassen. Dazu erstellt jetzt einen weiteren Wegpunkt, und dann noch einen in der Nähe des ersten. Diesem Gebiet ihr dann die Order Wiederholen.



Seht euch das an. Die Einheit läuft jetzt unbegrenzt oft zwischen den Wegmarken auf und ab. Diese Schleifen können natürlich auch mit mehreren Wegpunkten aufgebaut werden.

Es gibt viele Eigenschaften von Wegpunkten.

Bewegen: Einheit Bewegt sich zur Wegmarke

Zerstören: Einheit Rückt vor zum Ziel und zerstört

Einsteigen: Diese Wegmarke wird auf ein Fahrzeug gelegt. Die Einheit steigt dann ein.

Suchen und zerstören: Lläuft automatisch einen kleinen Bereich um die Wegmarke ab und feuert auf alles was sie findet.

Anschließen: Synchronisieren sie diese Wegmarke mit der einer anderen Gruppe. Dann folgt diese der zweiten Gruppe.

Anschließen und führen: Das Selbe wie Anschließen. Die eigene gruppe übernimmt jedoch die Führung.

Aussteigen: Steigt an der Stelle aus dem Fahrzeug aus.

Widerholen: Beginnt bei der dichtesten Wegmarke erneut.

Laden: Wartet bis andere Einheiten eingestiegen sind.

Entladen: Entlädt alle im Fahrzeug.

Transport entladen: Entlädt alle nicht zur Gruppe gehörenden Einheiten.

Halten: Bleibt bei der Wegmarke stehen.

Aufklären: Startet ein Aufklären und verfolgt Gegner. Rückt sachte vor.

Bewachen: Hält die Position, und begibt sich bei Kontakt zu der Stelle.

Sprechen: führt einen unter Effekte definierten Satz aus.

Geskripted: Startet ein Skript. Geht auch in jeder anderen Wegmarke.

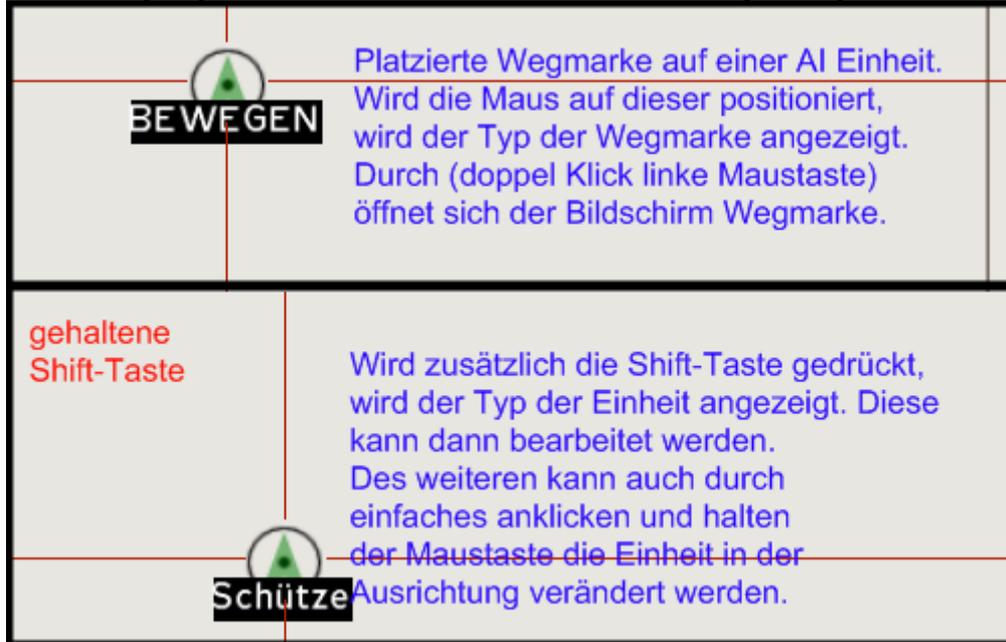
Unterstützen: Wartet an der Marke bis jemand Unterstützung anfordert.

In nächstes Einsteigen: Sucht das nächste Fahrzeug und steigt da ein.

Entlassen: Gruppe verteilt sich im Gelände. Setzt sich hin verteilt sich im Gelände.

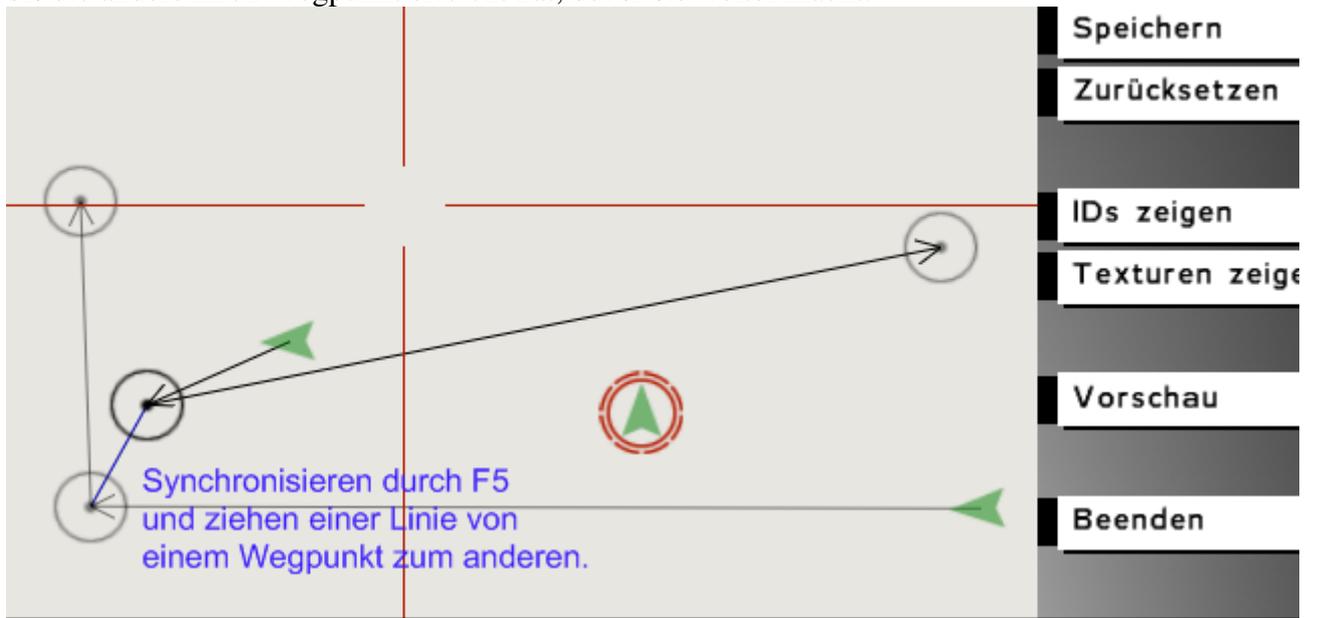
Einheit unter einer Wegmarke

Erstellt eine Ai Einheit und gebt dieser eine Wegmarke direkt auf der Einheit. Diese Methode wird heufig angewendet, um der Einheit Grundeinstellungen zu geben.



Synchronisieren

Nun wenden wir uns dem synchronisieren von Wegmarken zu. Dazu erstellt zwei unabhängige Einheiten mit je drei Wegmarken. Wählt den Punkt Synchronisieren durch Auswahl oder durch F5 und zieht eine blaue Linie von der ersten Wegmarke der ersten Einheit zur Wegmarke der zweiten Einheit. Jetzt sind die beiden Wegmarken auf einander abgestimmt. Wenn ihr die Mission ausprobiert könnt ihr beobachten, dass die Einheit wartet bis die andere ihren Wegpunkt erreicht hat, bevor sie weiter macht.



Auslöser

Nun wollen wir einen Auslöser mit in die Mission einbringen. Erstellt eine Spielereinheit. Danach erstellt eine zweite Einheit gebt ihr den Namen S1.

Dazu wählt Auslöser, F3 aus, erstellt einen Auslöser auf der Karte. Achse 0/0 Aktivierung keine. Und fügt unter Aktivierung **not alive S1** ein. Dadurch wird der Auslöser aktiv sobald S1 tot ist. Unter den Zeiteinstellungen Min 5, Max 15, Mid, 10 bedeutet das der Auslöser nicht sofort auslöst. Sondern noch 5 bis 15 Sekunden pausiert. Danach wird der Auslöser Bei Akt. hint "der Auslöser ist aktiv\nS1 ist tot" ist der Aufruf eine Textmitteilung aus zu geben.

Auslöser hinzufügen

Achse A Achse B

Winkel Ellipse Rechteck

Aktivierung Einmal Mehrfach

Countdown

Min.: Max. Mid.:

Typ

Name: Text

Bedingung

Bei Akt.

Bei Deak.

Effekte

Ai Einheit
Name S1

Spieler

Mission weiter bearbeiten

Wenn ihr euch eine Mission geladen habt und diese weiter zu bearbeiten besteht die Möglichkeit die Missions PBO einfach zu endpacken und weiter zu bearbeiten. Hierzu ist ein PBO Endpacker notwendig. Die endpackte Mission wird dann einfach unter eurem Benutzernamen in den Missionsordner verschoben. Danach ist diese im Editor zu finden.

Mission beenden.

Erstellt einen Auslöser mit der Spezialisierung Ende1 etc. Gebet dem Auslöser eine Variable welche erfüllt ein muss, oder definiert die Bedingungen.

Beispiel, die Mission soll enden wenn die Einheit John1 tot ist.

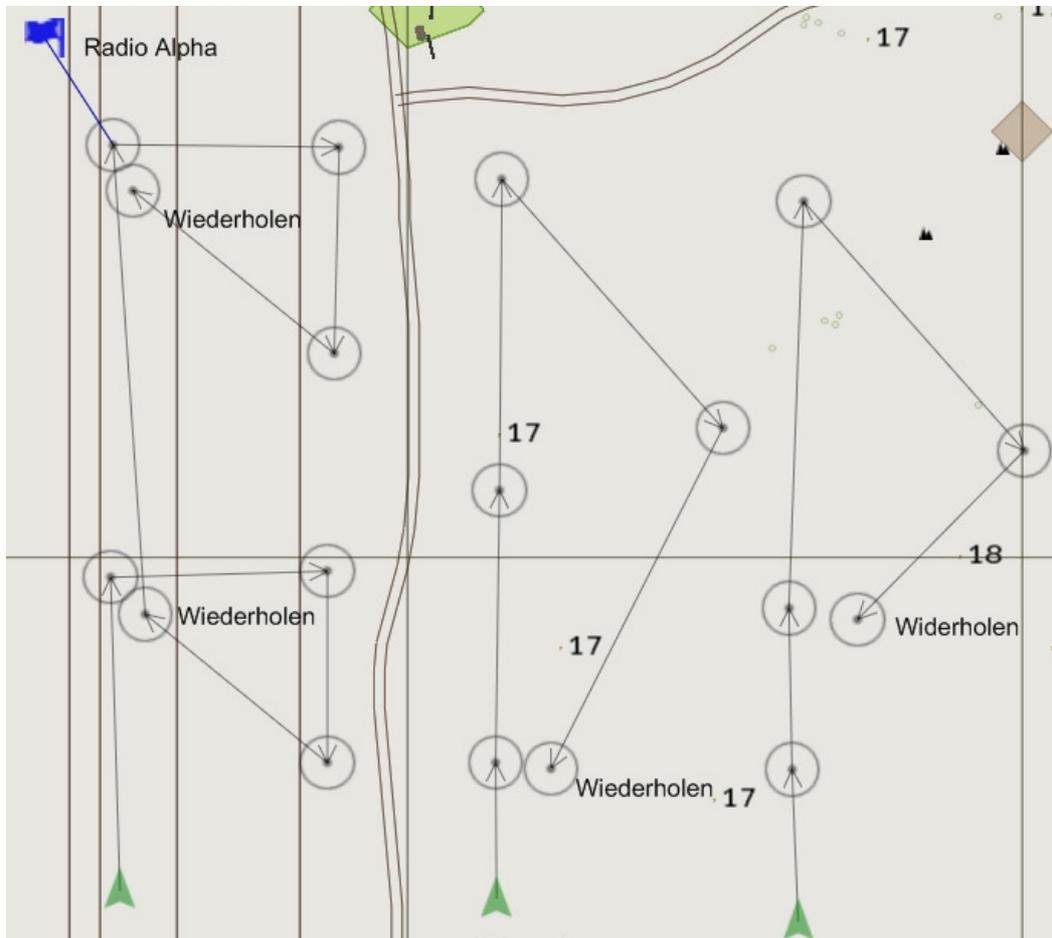
Dazu einen Auslöser erstellen, Ende 1 wählen. Sobald die Einheit jetzt tot ist, Endet das Spiel.

<p>Achse AB 0 Aktivierung keine. Unter Bedingung ist not alive John1 Was dann dafür sorgt, das wenn die Einheit John1 tot ist, der Typ Ende 1 angesprochen wird. Die Mission endet dann, und das Briefing mit der Benennung Ende1 wird angezeigt. (Natürlich nur wenn dieses Briefing existiert.)</p>	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>Achse A <input type="text" value="0"/></p> <p>Winkel <input type="text" value="0"/></p> <p>Aktivierung <input type="text" value="Keine"/></p> <p><input type="checkbox"/> Vorhanden Von Westen entdeckt Von Widerstand entdeckt</p> <p>Countdown <input type="text" value="0"/> Min. <input type="text" value="0"/> Max. <input type="text" value="0"/> Mid. <input type="text" value="0"/></p> <p><input type="checkbox"/> Timeout</p> <p>Typ Ende 1</p> <p>Name: <input type="text"/> Text <input type="text"/></p> <p>Bedingung <input type="text" value="not alive John1"/></p> <p>Bei Akt. <input type="text"/></p> <p>Bei Deak. <input type="text"/></p> </div> <div style="width: 45%;"> <p>Achse B <input type="text" value="0"/></p> <p><input type="text" value="0"/> Ellipse <input type="text" value="Rechteck"/></p> <p><input type="text" value="Keine"/> Einmal <input type="text" value="Mehrfach"/></p> <p><input type="checkbox"/> Nicht vorhanden Von Osten entdeckt Von den Zivilisten entdeckt</p> </div> </div> <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <input type="button" value="Effekte"/> <input type="button" value="OK"/> <input type="button" value="Abbrechen"/> </div>
--	---

Es können weitere Variationen für das Ende einer Mission erstellt werden. Dazu können dann weitere Punkte Ende 2 ,3 ,4 definiert werden. Je nach dem Stand der Mission können je nach Briefing verschiedene Texte im Abspann erstellt werden.

Schleifen laufen (patrouillieren)

Es gibt verschiedene Möglichkeiten Schleifen aufzubauen. Die Grundkonzeption ist jedoch immer gleich. Eine Kette von Wegpunkten wird wiederholt. Die Einheit arbeitet einen Wegpunkt nach dem anderen ab und wiederholt die Kette an der Wegmarke, die am dichtesten am Wegpunkt mit dem Typ „Wiederholen“ liegt. Es gibt auch die Möglichkeit, dieses mit Auslösern zu machen.



Beim rechten Beispiel liegt der Wegpunkt Wiederholen an der zweiten Wegmarke. Das bewirkt, dass der erste Wegpunkt nicht wiederholt wird.

Beim mittleren Beispiel liegt der Wegpunkt Wiederholen an der ersten Wegmarke. Das bewirkt, dass die ganze Kette wiederholt wird.

Beim linken Beispiel sind zwei Kreise aufgebaut. Wenn die Einheit startet, begibt sie sich über Wp1 Wp2 WP3 zum Wiederholen Wp4 und beginnt mit der Schleife erneut. Der obere Kreis WP5 WP6 WP7 WP8 ist am WP5 mit einem Funkauslöser synchronisiert, der auf folgende Weise eingestellt wird:

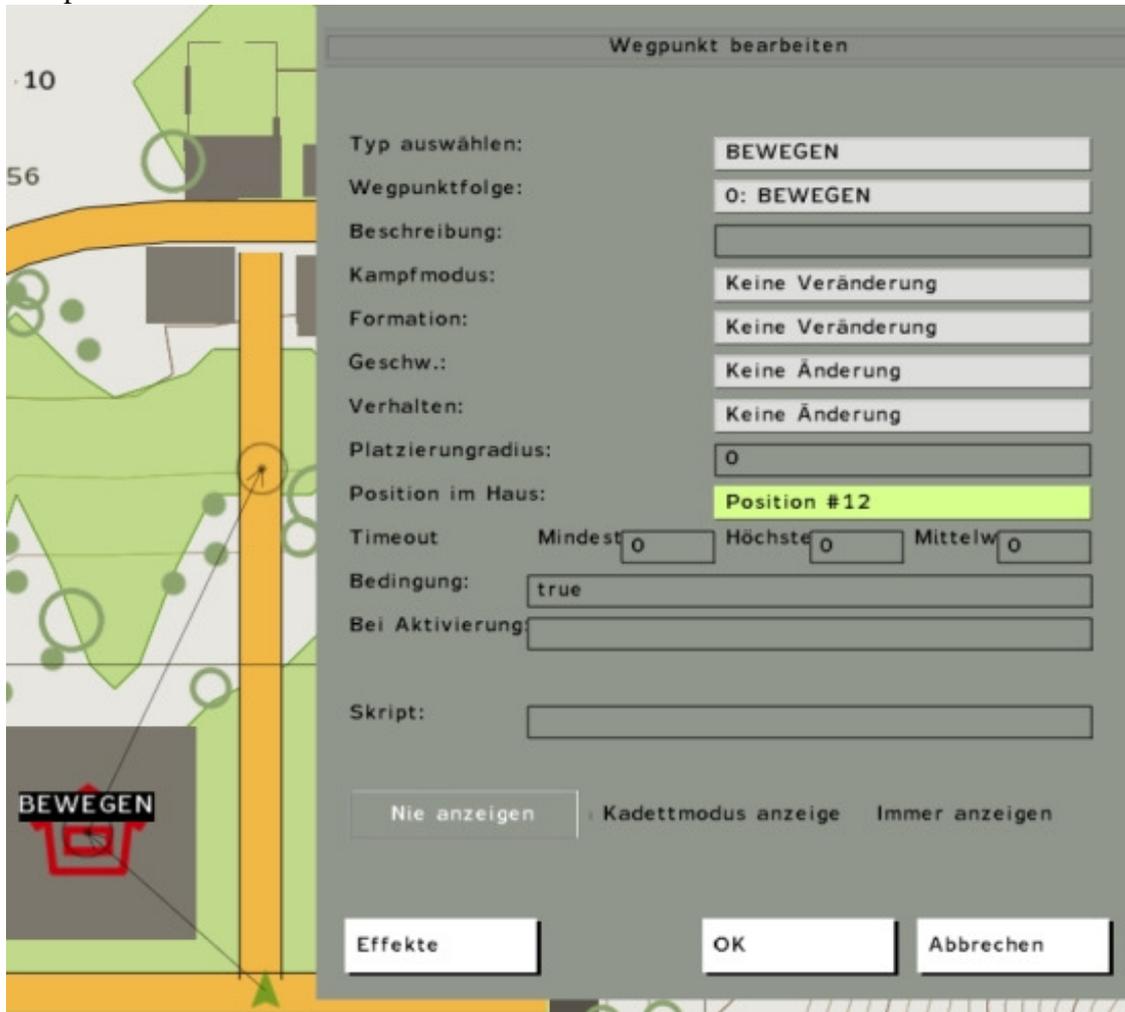
Achse AB: 0
 Aktivierung: Radio Alpha
 Typ: Schalter

Wenn jetzt über diesen Funkauslöser der obere Kreis gewählt wird, begibt sich die Einheit zu WP5 und wiederholt den Kreis WP5 WP6 WP7 WP8. Auf diese Weise können Einheiten mit verschiedenen Routen ausgestattet werden.

Einheit geht in ein Gebäude

Wenn man möchte, dass eine Einheit in ein Gebäude geht, legt man einfach einen Wegpunkt auf das Gebäude und bestimmt im Wegpunkt die Position, die die Einheit dort einnehmen soll. Der Weg zur Position an sich kann im Gebäude jedoch nicht weiter vorgegeben werden. Der Wegpunkteditor wird in dem Punkt Position im Haus erweitert. Da kann dann die Position im Haus eingestellt werden, den die Unit einnimmt.

Beispiel:



Als Position wurde in diesem Fall die Position 12 gewählt. Nach Erreichen dieser Position wendet die Einheit, verlässt das Gebäude wieder und setzt danach die Wegpunktroute fort. Wenn eine Gruppe einen solchen Wegpunkt erreicht, kann dies zu Stau führen, da die Wege im Gebäude nur für eine Einheit bestimmt sind.

Helikopter

Helikoptertransport Abladen

Eine Gruppe und einen Helikopter platzieren. Dem platzierten Helikopter einen Namen geben, im Beispiel: blackhawk1

Dem Gruppenführer der Gruppe, die transportiert werden soll, wird in die Init Zeile geschrieben:

```
{_x moveincargo blackhawk1 } foreach units this
```

Damit befindet sich die Gruppe bereits im Helikopter, wenn die Mission startet.

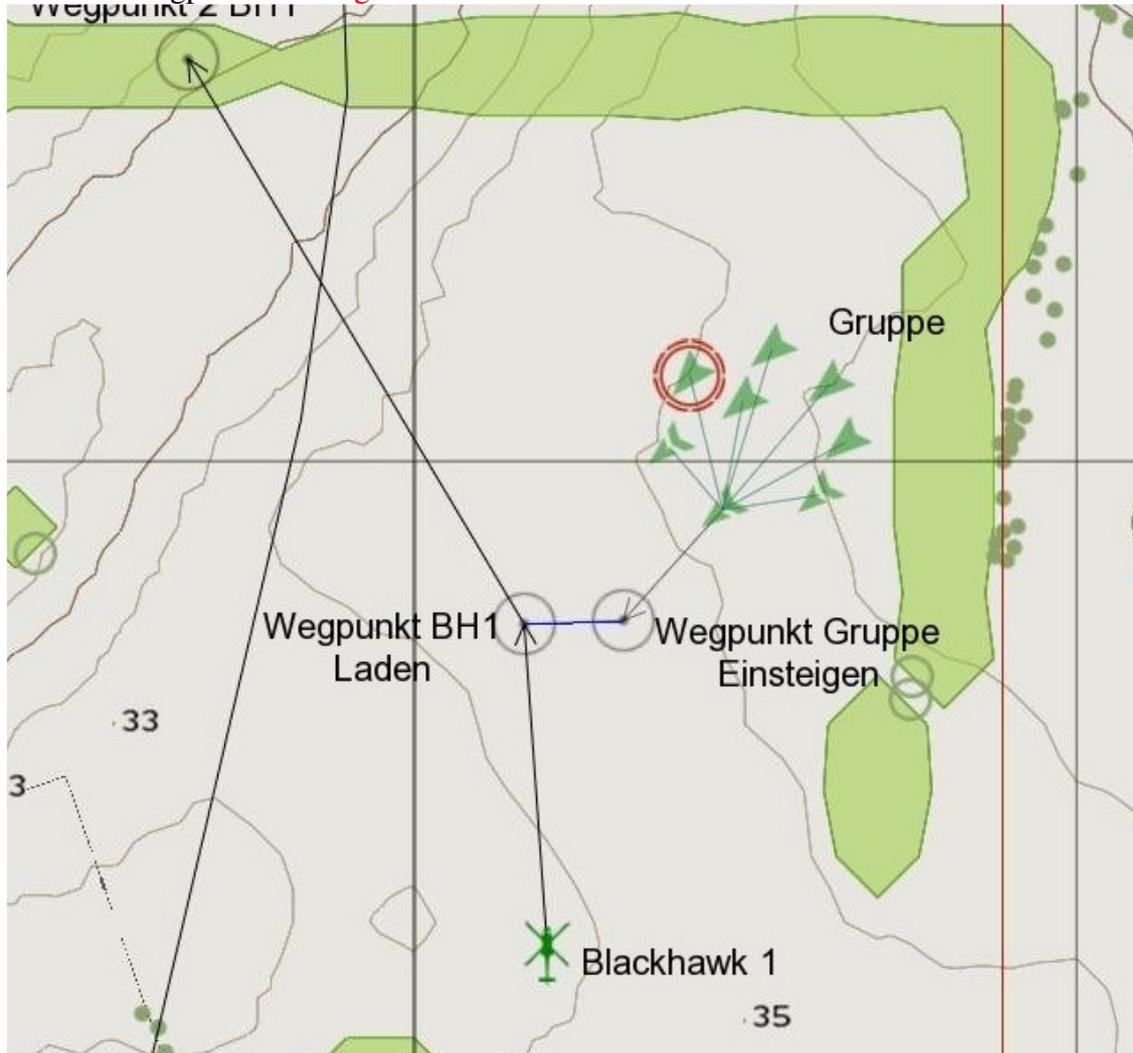
Der Helikopter erhält nun einen Wegpunkt "Transport Entladen", im Beispiel auf ein H, sodass der Helikopter an der vorgesehenen Stelle landet.



Gruppe 1	Sitzt im Helikopter durch den Befehl <code>{_x moveincargo blackhawk1 } foreach units this</code> in die Initzeile des Gruppenführers
Blackhawk1	Fliegt (Speziell: Fliegend) zum Wegpunkt Blackhawk1
Wegpunkt Blackhawk1	Transport entladen, der Blackhawk wird landen und die Gruppe aussteigen lassen
Wegpunkt Gruppe	Die Gruppe bewegt sich nach dem Aussteigen zu diesem Wegpunkt
H Zivil	Kann jedes beliebige H sein, wird benutzt, um den Helikopter an diesem bestimmten Punkt landen zu lassen. Der Wegpunkt blackhawk1 wird auf das H gelegt.

Helikoptertransport Abholen

Zuerst eine Gruppe und einen Helikopter platzieren. Der Helikopter erhält nun einen Wegpunkt "Laden", während die Gruppe einen Wegpunkt "Einsteigen" erhält, beide Wegpunkte werden mit Synchronisieren verbunden. Danach erhält der Helikopter einen weiteren Wegpunkt "Bewegen"



Gruppe	Wartet auf Abholung durch Blackhawk1
Blackhawk 1	Fliegt (Speziell: Fliegend) zum Wegpunkt Blackhawk1
Wegpunkt Gruppe Einsteigen	Gruppe bewegt sich zum Wegpunkt Einsteigen und wartet darauf, dass der Helikopter landet
Wegpunkt Blackhawk1 Laden	Blackhawk1 landet, sobald dieser gelandet ist, steigt die Gruppe ein.
Wegpunkt 2 Blackhawk1	Sobald die gesamte Gruppe eingestiegen ist, fliegt Blackhawk1 weiter zu diesem Wegpunkt.

Helikopterlandung

Es gibt verschiedene Wege, eine Lufteinheit zum Landen zu bewegen/zu landen. Der einfachste Weg ist, einen Wegpunkt zu setzen und diesen mit der Order "**Aussteigen**" zu versehen. Dies bewirkt, dass die Besatzung aussteigt.

Der etwas bessere Weg ist die in einem Wegpunkt eingetragene Order Name land "**land**". Dies bewirkt, dass der Helikopter landet, den Motor abstellt, jedoch niemand aussteigt. Die Order kann auch geändert werden in Name land "**getout**". Das bewirkt dann, dass der Helikopter landet und die Besatzung wieder aussteigt.

Alle diese Landungen haben jedoch das Problem, dass sie nicht punktgenau sind. Um dieses zu erreichen, setzt man den Wegpunkt "**Aussteigen**" auf ein Objekt oder erstellt hierfür einen Heliport (unsichtbar). Der Helikopter wird nun direkt diesen Punkt anfliegen und dort landen oder Einheiten absetzen.

Helikopter Abladen mal anders

Wenn es stört das eine AI beim landen so lange braucht kann das mit einem Trick umgangen werden. Hierzu wird eine Wegmarke auf ein Objekt (Heliport empty) gesetzt. Dieser wird mit einer globalen Variablen gesperrt damit der Helikopter nicht weiterfliegt. Über ein Skript wird jetzt die Flughöhe geregelt und über die Zeit wird jetzt das Aussteigen der Einheiten geregelt. Das sehen wir uns mal genauer an.

Erstelle einen Helikopter (Name Luft1), einen Heliport (Absetzpunkt) und eine Gruppe. Gib dem Anführer den Namen Gruppe1, und schreibe in dessen Init Zeile

{_x moveincargo Luft1} foreach units this . Damit diese zu Missionsbeginn gleich in der Einheit Luft1 sitzen. Jetzt legen wir einen Wegpunkt von der Einheit Luft1 auf das Objekt (Heliport empty) und geben unter Bedingung: abladenluft1 ein. Damit die Einheit nicht in ihrer Wegpunktkette weiter schreitet. Auf diesen Wegpunkt wird ein Auslöser gesetzt, welcher von der Einheit Luft aktiviert wird. Dieser startet ein Skript welches das Aussteigen regelt. Eintrag **[Luft1,Gruppe1,10,20,0.2,abladenluft1] exec "abladen.sqs"** .

Siehe zu diesem Thema Skripte und Fallschirm Skript.

Wenn die Einheit jetzt das Absetzen über einem Haus ausführen soll wird der dritte Parameter (in diesem Beispiel 10) auf die Höhe des Gebäudes gesetzt. Also auf freiem Feld auf 0 oder 1

<pre> ; ***** ; ** ArmA Script File ** ; ***** _air = _this select 0 _grp = _this select 1 _hoehe1 = _this select 2 _hoehe2 = _this select 3 _warten = _this select 4 _weiterflug = _this select 5 _air flyinheight _hoehe1 ~4 _aunits = units _grp _i = 0 _j = count _aunits #Schleife1 (_aunits select _i) action ["EJECT", _air] unassignvehicle (_aunits select _i) dostop (_aunits select _i) _i = _i + 1 ~2 ? _j > _i : goto "Schleife1" _weiterflug = true </pre>	<pre> ; ***** ; ** ArmA Script File ** ; ***** ;Parameter Name der Lufteinheit. ;Parameter name des Anführer Absetzgruppe. ;Absetzhöhe. ;Abflughöhe nach dem Absetzen. ;Zeit zwischen den units. ;Bedingung desWegpunktes. ;Befehl für _air auf Höhe _hoehe1 zugehen ;Skript pausiert 4 Sekunden ;Siehe Fallschirmskript ;setzt die Bedingung im Wegpunkt auf true. Luft1 fliegt weiter. (Ab v 1.05) ;Gibt den Befehl für _air auf Höhe _hoehe2 zu gehen. </pre>
---	---

Helikopter Einsammeln mal anders

Dies zeigt eine Möglichkeit die es erlaubt im MP Missionen ein Einsammeln durch eine AI zu gewährleisten welche unabhängig von Wegpunkten ist.

(Welcher Spieler hat schon Lust eine halbe Stunde in der Gegend zu kreisen bis er seine Kameraden wieder einsammeln soll).

Das heißt, dass der Spieler sich frei auf der Insel bewegt. Und keine Wegpunkte hat welche mit denen der Einheit synchronisiert wurden. In diesem Beispiel ist ein Abholplatz festgelegt, der über Funk Alpha aktiviert wird und den Helikopter anfordert. Dieser fliegt dann den Punkt an, geht dort in Warteposition und fliegt weiter nachdem alle an Bord sind. Dieses Beispiel ist für zwei Spieler geschrieben. Es kann natürlich auch erweitert werden.

Zuerst den Spieler (die Spieler) setzen (Unit1 und Unit2). Danach den Helikopter platzieren. Einen Marker auf der Karte platzieren, diesen beschriften (Landezone etc) und einen Wegpunkt für den Helikopter darauf platzieren. (einen weiteren der auf die Landezone führt). Die Bedingung für den Landewegpunkt setzen (all-in in diesem Beispiel). Weitergehend muss auf dem Startpunkt des Helikopters noch ein Wegpunkt mit der Bedingung abholen gesetzt werden, damit dieser nicht gleich zur Landezone fliegt. Auf den Marker wird noch ein

Auslöser welcher von dem Helikopter ausgelöst wird und ein Skript startet.

[Luft1,Unit1,Unit2,1,20,allin] exec "einsammeln.sqs" .

Skriptbeispiel:

<pre> ; ***** ; ** ArmA Script File ** ; ***** _air = _this select 0 _unit1 = _this select 1 _unit2 = _this select 2 _hoehe1 = _this select 3 _hoehe2 = _this select 4 _bedingung= _this select 5 _air flyinheight _hoehe1 #schleife ~1 ? (((_unit1 in _air) or (not alive _unit1))and((_unit2 in _air) or (not alive _unit2))): goto "abflug" goto "schleife" #abflug _bedingung = true _air flyinheight _hoehe2 </pre>	<pre> ; ***** ; ** ArmA Script File ** ; ***** ; Parameter auslesen ;gibt den Befehl für _air auf Höhe _hoehe1 zu gehen (Flughöhe) ;Schleife wird solange wiederholt bis unit1 und unit2 im Heli oder tot sind ; wenn Bedingung erfüllt fliegt der Heli ab </pre>
---	---

Expertenversion:

Über einen Einzelklick auf der Karte die Landezone bestimmen.

Hierfür bietet es sich an ganz ohne Wegpunkte zu arbeiten. Es werden die Truppen auf der Karte erstellt. Unit1, Unit2 ... Der Helikopter Luft1 und ein Marker Marker1 (dieser kann auch über Createmarkerlokal automatisch erstellt werden. v1.06)

Editieren

Einheit beschädigen

Um einer Einheit einen Schadenswert zu geben, einfach in die Init Zeile **this setdamage Wert** setzen. Das geht natürlich auch über **Name setdamage Wert**.

Beispiel : **John setdamage 1**;

Bewirkt, dass die Einheit John den Schadenswert 1 erhält und somit tot ist. Dies geht auch mit Objekten, die im Editor platziert wurden.

Beispiel : **(object 25328) setdamage 1**; (hier ist noch nicht klar was da los ist.)

Entfernung zweier Objekte / Units abfragen

Um die Entfernung von zwei Objekten/Einheiten zu bestimmen, kann der Befehl **Distance** verwendet werden. Name der variable = Name der ersten Einheit Distance Name der zweiten Einheit.

Beispiel : **_entfern = Alex Distance John**

_entfern ist die lokale Variable, in die der Wert gegeben wird. Dieser ist in Metern.

Auslöser versetzen

Um einen Auslöser zu versetzen, wird dieser durch den Befehl

Name des Auslösers setpos getpos Name der Einheit der Auslöser zu der Einheit versetzt.

Beispiel : **Trigger1 setpos getpos Alex**;

Aktion im Aktionsmenü eintragen

Das Aktionsmenü ist unten rechts im Bildschirm und erlaubt dem Spieler, Türen zu öffnen, Leitern zu benutzen, usw. Um in dieses Menü neue Einträge hinzuzufügen, wird über ein Skript, Init Zeile, Auslöser, etc. ein neuer Eintrag erstellt. Name des Eintrags, Name der Einheit Object, der den Eintrag bekommen soll = ["Aktion Name", "Name des Skriptes, welches gestartet werden soll"].

Beispiel : **Aktion1 = Alex addAction ["Ari anfordern", "dialog.sqs"]**;

Wenn das Menü jetzt gewählt wird, erscheint der Name „Text ausgeben“ als Auswahl und wenn es aktiviert wird, startet das Skript text.sqs (wenn es existiert). Um das Menü zu entfernen, wird der **Name der Einheit RemoveAction Name des Eintrags** aufgeführt.

Beispiel : **Alex RemoveAction Aktion1**

Fahne am Mast

Es besteht die Möglichkeit, eine eigene Fahne zu nutzen. Hierzu wird ein Fahnenmast platziert, in dessen Init Zeile **this SetFlagTexture "Fahne1.jpg"**; geschrieben. Das Format sollte 1 zu 2 sein. Es bietet sich an 128 zu 256 zu nehmen. Das Format kann *.jpg oder *.paa sein. Das Bild der Fahne muss im Missionsordner hinterlegt werden. Die besten Fahnenmaße sind n zu 2n

Schranke öffnen – schliessen

Um eine Schranke zu bedienen wird unterschieden zwischen einer im Editor platzierten und einer auf der Insel fest installierten. Vorgehensweise für eine im Editor platzierten Schranke.

Um dieses Objekt an zu sprechen wird dieser ein Name gegeben. Dieser kann dann über **(Name der Schranke) animate ["Bargate", 1/0]**; geöffnet oder geschlossen werden.

Das 1/0 bestimmt ob die Schranke offen oder geschlossen sein soll.

Ist die Schranke auf der Insel verbaut muss diese zuerst mit einer Logik erfasst werden. Dazu

neben die Schranke eine Logik stellen (Name Logik1) und die ID der Schranke eintragen.
 (getPos Logik1 nearestObject 175399) animate ["Bargate", 1/0];

Objekte / Einheiten schweben lassen

Um Einheiten oder Gegenstände wie Schießscheiben oder Tische entweder schweben zu lassen oder weiter im Boden zu versenken, einfach den Befehl in der Init Zeile des Gegenstandes eintragen.

Beispiel : **this setpos [(getpos this select 0),(getpos this select 1),10];**

Die Angabe 10 steht für die Höhe und ist in Metern. Vorsicht bei dem Platzieren von Einheiten. Diese können sich schnell die Beine brechen, wenn sie aus einer zu großen Höhe fallen. Im MP gibt es mit diesem befehl Probleme.

Erstellen neuer Einheiten

Um eine neue Einheit zu erstellen, kann der Befehl

Name der Einheit = "Typ der Einheit" CreateVehicle [X Koordinate, Y Koordinate] verwendet werden.

Beispiel : **Neu1 = "UH60" CreateVehicle [125485,142574]**

Der X und Y Wert sind die Längen und Breiten auf der Karte. Diese kann man sich aus der mission.sqm entnehmen, indem man eine Testeinheit erstellt. Leichter ist es daher, eine Einheit bei einem Marker oder anderen Objekt zu erstellen.

Beispiel : **Neu2 = "UH60" CreateVehicle Getpos Heliport1**

Heliport1 ist der Name eines Heliport, welcher im Voraus auf der Karte platziert wurde.

Einheit löschen

Um eine Einheit im Spiel zu löschen wird **DeleteVehicle Name der Einheit** verwendet.

Beispiel : **DeleteVehicle car1**

Es ist leider nicht möglich die auf der Insel platzierten Objekte zu löschen.

Um einen Panzer zu löschen kann dieser Befehl auch verwendet werden. Jedoch bleiben dann die drei Besatzungsmitglieder erhalten. Mit dem Befehl **deleteCollection tank1** wird das Fahrzeug mit allen Insassen gelöscht. (tank1 ist der Name des Panzers)

Sichtweite im Spiel anpassen

Mit dem Befehl **SetViewDistance 1000;** ist es möglich, die Sichtweite im Spiel zu ändern.

Damit kann zum Beispiel die Sichtweite reduziert werden, wenn der Spieler in ein Nebelgebiet kommt. Es macht Sinn, dieses im MP Missionen zu machen, wenn die Servereinstellungen für diese Mission geändert werden sollen.

Zeitmaschine

Die Spielgeschwindigkeit kann durch den Befehl **SetAccTime Wert** geändert werden. Somit kann in einem Intro ein Sonnenaufgang in wenigen Sekunden gezeigt werden oder eine Mission beschleunigt werden.

Beispiel : **SetAccTime 1.5**

Das geht natürlich auch in andere Richtung

Beispiel : **SetAccTime 0.5**

SetAccTime geht nur im Singleplayer.

Weitergehend besteht die Möglichkeit, dem Spiel eine Zeit zu geben.

Befehl : **SkipTime 8**

Hierbei wird der eingestellten Spielzeit 8 Stunden hinzu addiert. Wird ein negativer Wert

eingegeben wird die Uhr zurückgestellt. Dies macht Sinn, um ein Intro und die Mission zeitlich voneinander zu trennen.

Marker sichtbar machen

Voraussetzung ist, dass der Marker im Voraus auf unsichtbar gesetzt wurde. Dies kann in der Init Zeile erfolgen.

Beispiel : "Name des Markers" **setmarkertype** "empty";

Wird jetzt über einen Auslöser Skript dem Marker ein Typ gegeben, wird dieser sichtbar.

Die Standard-Marker :

Start, Objektive, Flag, Dot, Destroy, Start, End, Warning, Join, Pick Up, Unknown, Marker, Arrow

Beispiel : "Name des Markers" **setmarkertype** "Pick Up";

Marker: "Flag" "Flag1" "Dot" "Destroy" "Start" "End" "Warning" "Join"
"Pickup" "Unknown" "Marker" "Arrow" "Empty"

Inselkarte auf den Monitor bringen

Es macht in einigen Bereichen Sinn die Karte auf den Monitor zu erzwingen. Das ist dann sinnvoll, wenn zum Beispiel eine Route dem Spieler mit Markern gezeigt werden. Befehl **ForceMap true** oder **ForceMap false**, um die Karte wieder zu entfernen.

Eingaben des Spielers abschalten

Um ungeduldigen Spielern die Möglichkeit zu nehmen, Intros abubrechen oder vorzustürmen, kann die Eingabe von der Tastatur gesperrt werden. Dies erfolgt über den Befehl **DisableUserInput True**; Um die Tastatur wieder frei zu schalten **DisableUserInput False**

Bestimmten MP player ansprechen

Wenn ihr jemanden in einer Mission direkt ansprechen wollt geht diesem am leichtesten über den Namen. Es besteht die Möglichkeit diesen abzufragen und daraufhin eine Aktion folgen zu lassen. Wenn ihr diesen ärgern wollt, kann ihm der Schadenswert 0.9 gegeben werden. Dann darf dieser erst einmal zum Sanni kriechen. So etwas sollte jedoch nicht übertrieben werden. Befehl im Skript oder im Auslöser:

if ((! local player) or (name player != "Alex.Sworn")) then { exit; }; Dieser Aufruf sorgt dafür das wenn der Player Alex.Sworn heißt die Aktion in der nächsten Zeile ausgeführt wird.

Wenn nicht wird das Skript beendet.

Einheiten in einem Bereich ansprechen

Einheiten können in Listen erfasst werden, Befehle Aktionen können somit an alle Einheiten in einem Bereich übergeben werden. Befehle wie Waffen entfernen, Verhalten von Einheiten können so einfach ausgeführt werden. Oder es besteht auch die Möglichkeit das Einheiten in einem Bereich gezählt werden. Usw. Hierzu wird ein Auslöser benötigt welcher von der jeweiligen Seite angesprochen wird. Diesem Auslöser gebt ihr jetzt einen Namen, bei Aktivierung werden jetzt alle Einheiten in dem Bereich in die Liste (Name des Auslösers) eingetragen. Diese Liste kann jetzt nach eigenem Ermessen genutzt werden. zB.

{_Variable Befehl} foreach Name der Liste

Feindgebiet ist Feindfrei

Um sicherzustellen dass ein Feindgebiet (Dorf mit Feindeinheiten) feindfrei ist, gibt es 3 Möglichkeiten dies zu überprüfen:

1. Auslöser erstellen mit der Bedingung: Feindseite nicht anwesend. In Init des Auslösers eintragen: hint "Gebiet ist Feindfrei"

2. Alle Feindeinheiten mit Namen benennen. zB „Ost1, Ost2...“. Auslöser erstellen – Bedingung Ost nicht anwesend. In Init des Auslöser schreiben:

```
("alive_x" count [Ost1,Ost2]) = 0
```

Wenn alle Einheiten Tod sind wird nun der Auslöser aktiviert.

3. Da oftmals mehrere Feinde fliehen und es unter Umständen sehr lange dauern kann bis das Gebiet geklärt ist, eignet sich folgende Methode gut: Gruppen erstellen, in die jeweilige Init Zeile der Gruppenführer eintragen: Ostgroup1 = group this; Ostgroup2 = group this. Auslöser erstellen: Bedingung:

```
"alive_x" count (units(Ostgroup1) + units (Ostgroup2)) <= 5
```

Erst wenn die Truppenstärke des Feindes unter 5 Mann gesunken ist wird nun der Auslöser aktiviert. Diese Methode ist für große Gebiete geeignet und um das Spiel nicht ein halbe Ewigkeit dauern zu lassen

Todeszone

Dieses Kapitel beschäftigt sich mit der Zerstörung von Einheiten in einem Bereich. Hierzu wird ein Auslöser erstellt, Größe Form ist hierbei egal. Jetzt besteht die Möglichkeit diesem Auslöser einen Namen zu geben (zB.Todeszone1). In einem Skript etc kann jetzt diese Liste angesprochen werden. `{_x SetDamage 1} forEach list Todeszone1` oder es besteht die Möglichkeit dies gleich zu tun

```
{_x SetDamage 1} foreach thislist
```

Einheiten in einem Bereich zählen

Um Einheiten zu zählen gibt es verschiedene Möglichkeiten.

Möglichkeit über ein EventHandler: Zuerst in jede Initzeile der zu Zählenden Einheit `this AddEventHandler ["killed",{_this exec "zaeler.sqs"}]` eintragen. Jetzt wird jedes mal wenn eine der Einheiten verstirbt ein Skript ausgerührt. Name zaeler.sqs dieser Name ist frei gewählt. In das Skript wird jetzt nichts anderes als zu einer Variable eins dazu gezählt. Die Variable sollte `global` sein da sie ansonsten nicht nutzbar ist.

zaeler.sqs

```
Anzahl = Anzahl +1
```

```
Exit
```

Wenn jetzt ein Ereignis wie (5 Soldaten getötet) etwas auslösen soll kann einfach die Variable in einem Auslöser abgefragt werden. ZB. Bedingung Anzahl = 5 und bei Aktivierung hint“5 Tote“ Jetzt erscheint eine Meldung wenn 5 Leute getötet wurden.

Möglichkeit über eine Liste

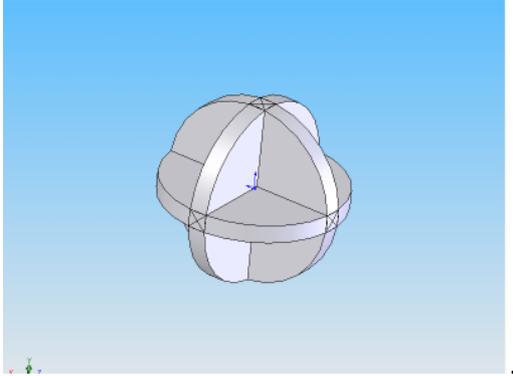
Siehe Einheiten in einem Bereich ansprechen. Das Ganze funktioniert genau wie in der Liste. Lediglich wird ein Zähler eingebaut. `i = i + 1` Dieser kann dann abgefragt werden.

Oder als kleine Spielerei die Funkspruch Methode.

```
Player sidechat format["%1",list Name des Auslösers oder der Liste];
```

Objekte schräg platzieren

Ein kleines aber nützliches Problem ist das Objekte welche im Editor Platziert werden sich am Gelände orientieren. Wenn das begradigt oder verstärkt werden soll wird einfach in die Init Zeile des Objekt folgendes geschrieben: `this setvectorUP [0,0,0.01]` oder diese Werte werden ja nach Wunsch angepasst. Als Richtwer ist hier dir Sinus zu nehmen. Also der Wertebereich -1 bis 1. Vorsicht diese Werte sind nicht an das Objekt gebunden sondern immer nach Norden ausgerichtet. Sie beziehen sich daher je nach Winkel des Objektes in alle Richtungen. Oben unten oder auch rollen.



Drei Winkel drei Linien. Darüber ist jeder Körper im freien Raum auszurichten. Der Umgang ist jedoch etwas schwerer.

Objekte auf der Achse drehen

Um ein Objekt zu drehen besteht die Möglichkeit die über `setDir` zu machen. Etwas eleganter ist da die Methode das über Name der Einheit `setvector Dir [sin _a,cos _a,0]` bei 0° Ausrichtung und keiner Rollauslegung. `_obj setVectorDIR [sin _a, cos _a, 0];` Objekte können unter Verwendung dieser Berechnung gewendet und gedreht werden.

```
_vx = (sin _b)*(cos _p);
_vz = -(sin _p);
_vy = (cos _b)*(cos _p);
_v = [_vx,_vz,_vy];
_obj setVectorDIR _v;
```

Objekte auf der Achse kippen

Um ein Objekt zu kippen. `_obj setVectorUP [sin _a, cos _a, 0];`

```
_vx = (sin _b)*(cos _p);
_vz = -(sin _p);
_vy = (cos _b)*(cos _p);
_v = [_vx,_vz,_vy];
_obj setVectorUP _v;
```

Speichern

ArMA bietet dem Missionsbauer verschiedene Möglichkeiten Spielstände, Zustände von Einheiten, Zustände von Spielern zu speichern. Diese werden extern vom Spiel abgelegt und können auch nach einem Neustart abgerufen werden.

Mission speichern

Um einen Missionstand zu speichern kann der Spieler durch Drücken der ESC Taste ein Menü aufrufen und dort den Button Speichern betätigen. Es besteht jedoch auch die Möglichkeit dies über ein Skript oder durch einen Auslöser zu machen. Dazu muss man in das Skript einfach den Befehl **savegame** schreiben. Bei Erreichen dieses Befehls speichert das Spiel automatisch den Spielstand. Stirbt der Spieler danach hat er die Möglichkeit zu dem Zeitpunkt zurück zukehren. Dies ist nützlich wenn man einige Missionsziele nacheinander erfüllen möchte. Dadurch braucht der Spieler die Mission nicht immer von Beginn an neu zu spielen.

Wenn eine Mission nur einen Speicherpunkt erhalten soll wenn sie im Cadet Modus gespielt wird. Quellcode

```
? (cadetMode) : savegame;
```

Missiondaten erfassen / speichern

Um einen Status eines Players zu speichern, damit er in einer anderen oder nächsten Missionsorder abgerufen werden kann, hat der Missionsbauer die Möglichkeit das mit den Befehlen saveStatus und loadStatus zu bewältigen.

```
player saveStatus "playerState"
```

speichert den Status des Players in der Variable playerState

```
player loadStatus "playerState"
```

liefert den Status des Players aus der Variable playerState

Playerstatus über ID erfassen

Desweiteren ist es möglich, verschiedene Playerdaten (abhängig von der ID) zu erfassen. Somit kann dann ein Spieler seinen alten Status auch nach einem Absturz und einem ArMA Neustart zurück erhalten. Und das unabhängig davon welche Figur er beim neu verbinden wählt. Seine Daten bleiben im Speichern von ArMA erhalten.

Befehl:

```
player saveIdentity "playerIdentity"
```

zum Speichern, und

```
player loadIdentity "playerIdentity"
```

zum Laden.

Variablen Speichern

Es können auch Variablen gespeichert werden. Wenn in Mission1 ein Ziel zu erfüllen ist (zB eine Person zu töten) und dieser Punkt erfüllt ist, kann dies eine Variable setzen, welche gespeichert wird. In der nachfolgenden Mission kann dann diese Variable abgefragt werden, und je nach dem kann diese Variable dann Aktionen starten. Dadurch werden Missionen mehr ineinander verschachtelt.

Befehl:

```
saveVar "name"
```

Speichert die Variable.

In dem folgenden Spiel einfach die Variable abfragen. Die gespeicherte Variable ist global

abrufbar. Dies geht jedoch nur in einer Kampagne und der Zustand ist nur in den Nachfolgenden Missionen aus zu lesen.

Speicher löschen

Sollte der Spieler nicht wollen das ein Speicherstatus nach Neustart der selben Mission noch erhalten ist sollte, muss dieser Speicherpunkt oder die gespeicherte Variable gelöscht werden. Es empfiehlt sich alle verwendeten Speichervariablen eindeutig zu benennen damit keine Überschneidungen mit anderen Missionen passieren. Denn wenn die Variable alert auf true gesetzt ist und gespeichert wird, ist das ein Problem für alle anderen Missionen welche mit der Variable arbeiten.

Um den Status des Spielers zu löschen:

`deleteStatus "playerState";`

einfügen. Um die Identität des Spielers zu löschen

`deleteIdentity "playerIdentity"`

einfügen.

Funkverkehr

Funksprüche

Die Texteinblendungen, die unten links im Bildschirm auftauchen.

Beispiel-Nachricht vom HQ : `[west,"HQ"] sidechat "Funkspruch";`

West ist die Seite, HQ steht für CROSSROAD.

Globale Funkmitteilungen

Beispiel : `Name GlobalChat "Funkspruch"`

Gruppenfunk

Beispiel : `Name GroupChat "Funkspruch"`

Eigene Seite anfunken

Beispiel : `Name SideChat "Funkspruch"`

Gruppe ein Rufzeichen geben

Um eine Rufzeichen einer Gruppe oder Einheit zuzuweisen, wird der Befehl

`Name der Einheit Gruppe setGroupID ["Delta","GroupColor4"]` verwendet.

Es gibt als Namen diese Möglichkeiten :

"Alpha" "Bravo" "Charlie" "Delta" "Echo" "Foxtrot" "Golf" "Hotel" "Kilo" "Yankee" "Zulu"

"Buffalo" "Convoy" "Guardian" "November" "Two" "Three" "Fox"

und als Farbe die Möglichkeiten :

"GroupColor0" - (Nothing) "GroupColor1" - Black "GroupColor2" - Red "GroupColor3" -

Green "GroupColor4" - Blue "GroupColor5" - Yellow "GroupColor6" - Orange

"GroupColor7" - Pink "Six" - Six

Beispiel : `group1 setGroupId ["Delta","GroupColor4"]`

Textmitteilungen

Mit dem Befehl `hint` können Textmitteilungen im oberen linken Bereich angezeigt werden. Es

gibt zwei Versionen dieses Befehls, `hint` erzeugt eine Messagebox, das Spiel läuft einfach

weiter. `hintc` erzeugt eine Messagebox die das Spiel stoppt. Diese erscheint in der Mitte des

Bildschirms. `hintCadet` ist eine Messagebox, die nur im Kadett-Modus angezeigt wird. Diese

Messagebox pausiert das Spiel ebenfalls und erscheint auch in der Mitte.

Beispiel : `hint "Der Weg ist frei";`

Es besteht die Möglichkeit, Text und Variable zu mischen.

Beispiel : **hint format** ["Der erste Wert ist %1\nDer zweite Wert ist %2",_wert1,_wert2];
 Hierbei wird der Befehl **hint format** verwendet. das Zeichen \n erzeugt einen Zeilenumbruch.
 Das Zeichen %1 und %2 sind Platzhalter für die nach dem Komma definierten Variablen.
 Eine weitere Form der Textmitteilungen ist

Beispiel :

TitleCut [format["erste Wert %1\nzweite Wert %2",wert1, wert2], "PLAIN DOWN", 1]

AI / KI System

Einheit plazieren

Die AI lässt sich bei Einheiten an oder ausschalten. Dadurch wird es möglich, Einheiten dazu zu bewegen, sich nicht hinzulegen, zu bewegen oder zu zielen. Weitergehend ist es möglich, eine Reihe von nützlichen Erweiterungen des AI-Verhalten abzustellen. Zum Beispiel ein Scharfschütze im Busch soll liegen und sich nicht bewegen.

Befehle:

Einheit DisableAI "autotarget"	stop the unit to watch the assigned target
Einheit DisableAI "target"	prevent the unit from assigning a target independently and watching unknown objects
Einheit DisableAI "move"	disable the AI's movement
Einheit DisableAI "Anim"	disable ability of AI to change animation.

Nach Speichern und erneutem Aufruf der Mission ist dieses allerdings aufgehoben.

Nach Platzierung einer Einheit im Editor unter Verwendung der Einstellung: „Spezial: keine“ würde die Einheit wieder in Formation laufen. Um das zu verhindern, trägt man „dostop this“ in die Init Zeile der Einheit ein. Dadurch bleibt die Einheit an der ihr im Editor zugeteilten Position stehen. Um das wieder rückgängig zu machen, schreibt man **Name dostop false**.

Gruppe Starten / Stoppen

Um eine Gruppe starten oder zu stoppen, wird der Befehl „Anführer der Gruppe stop true“ verwendet und um die Gruppe weiter ziehen zu lassen, der Befehl „Anführers der Gruppe stop false“ gesetzt. Dies kann über ein Skript mit einer Distanzabfrage kombiniert werden, so dass sich Gruppen in einem Waldstück nur dann bewegen, wenn sie keinen Feind in der Nähe haben.

Körperhaltung der Einheit

Um zu verhindern, dass eine Einheit sich hinlegt oder aufsteht, stehen die Befehle **this SetUnitPos "up"**; zur Verfügung.

Es gibt die Befehle:

up	stehen
Middle	kniet
down	liegen
auto	schaltet das Verhalten auf Automatik.

Animation einer Einheit

Dieses Kapitel beschreibt eine Möglichkeit Animationen für das Missionsumfeld zu erzeugen. Diese werden in zwei verschiedene Gruppen geteilt. switchmove und playmove.

Befehl: **Name der Einheit playmove "Animationsbefehl"**

Befehl: **Name der Einheit switchmove "Animationsbefehl"**

Im Inhaltsverzeichnis ist unter Datenbank auch eine Animationsliste zu finden.

Blickrichtung der Einheit

Wenn eine Einheit eine andere ansehen soll, verwendet man den Befehl

Name der Einheit1 doWatch Name der Einheit2.

Beispiel : **Alex DoWatch Brüste**

Jetzt sieht die Einheit mit dem Namen Alex auf die Einheit mit dem Namen Brüste.

Blickrichtung und Ausrichtung

Name der Einheit setDir 220

Einheit richtet sich in Richtung 220

Name der Einheit setDir getdir Name der Einheit2

Einheit1 richtet sich in Richtung der Einheit2

Name der Einheit doWatch Name der Einheit2

Einheit1 richtet sich in Richtung der Einheit2.

Name der Einheit lookAt Name der Einheit2

Einheit1 richtet sich in Richtung der Einheit2

Blickrichtung aufheben

Wenn eine Einheit mit dem dowatch Befehl auf eine andere ausgerichtet wurde ignoriert diese leider alles andere. Um das zu verhindern kann in einem Auslöser Skript der Aufruf

Name der Einheit dowatch objNull erfolgen. Damit richtet sich die Einheit wieder normal aus.

Einheit feuert auf Einheit

Wenn eine Einheit auf eine andere schießen soll, läuft das wie beim DoWatch Befehl. Befehl

Name der Einheit1 DoFire Name der Einheit2.

Beispiel : **Alex DoFire John**

Jetzt schießt die Einheit mit dem Namen Alex auf die Einheit mit dem Namen John.

Wenn man möchte, dass eine Waffe abgefeuert wird, kann dies auch über den Befehl Name fire "Waffen Namen"

Beispiel : **Alex fire "G36A"**

oder Beispiel: **Soldat1 fire ["pipebombmuzzle","pipebombmuzzle","pipebomb"]**

für eine Bombe.

Gesichtsausdruck

Einheiten können auch verschiedene Gesichtsausdrücke annehmen. Befehl

Name der Einheit SetMimic "Name der Emotion".

Beispiel : **Soldier1 SetMimic "angry";**

Es gibt die Emotionen normal, angry, agresive, cynic, smile, surprise, hurt, sad, ironic

Einheiten in Vehicle

Ein Skript muss nicht eine externe Datei sein. Es reicht schon aus, wenn einzelne Befehle in der Init Zeile oder in einem Wegpunkt aufgerufen werden. Damit kann zum Beispiel eine bestimmte Einheit als Driver, Gunner, Commander bestimmt werden oder eine ganze Gruppe in den Laderaum eines LKWs gesetzt werden. Um das zu erreichen, erstellen wir uns einen leeren Panzer mit dem Namen Tank1 und drei Soldaten mit den Namen S1 S2 S3. Bei der Einheit S1 schreiben wir in die Init Zeile

S1 moveincommander Tank1; S2 moveingunner Tank1; S3 moveindriver Tank1;

Das führt jetzt dazu, dass die drei Einheiten in der Commander Gunner Driver Position landen. Oder um die cargo Position zu beladen **Name1 moveincargo Name2**

Gruppe im Laderaum eines LKWs / Helikopters / Boot etc.

Um eine Gruppe in den Laderaum eines LKW zu bekommen, können wir sie über einen Wegpunkt einsteigen lassen oder jedem Gruppenmitglied den Befehl `this moveincargo LKW1` schreiben. Wobei LKW1 der Name des Transporters ist. Aber das geht auch einfacher.

Wir nennen den Anführer der Gruppe LKWGruppe1 und schreiben in dessen Init Zeile `{_x moveincargo LKW1} foreach units LKWGruppe1;`

LKW1 ist der Name des LKWs.

Wenn eine Einheit nicht in ein Fahrzeug, sondern mitlaufen soll, kann eine Einheit für den Transporter auch gesperrt werden. Dazu wird in die Init Zeile dieser Einheit

`this allowGetIn false` geschrieben. Wenn es sich um mehr als eine Einheit handelt, kann auch `[S1,S2,S3,S4]allowGetIn false` geschrieben werden. Um das wieder rückgängig zu machen,

wird in einem Skript, Auslöser, Wegpunkt einfach `[S1,S2,S3,S4]allowGetIn true` aufgerufen.

Der nächste Schritt ist jetzt die Abfrage, ob eine Einheit in einem Vehicle ist. Dies ist dann nützlich, wenn ein Spieler von einem Auto etc an einer bestimmten Position abgeholt werden soll.

Dazu kann in einem Wegpunkt die Bedingung `player in LKW1` oder `S1 in LKW1` geschrieben werden. LKW1 ist der Name des Vehicle und S1 der Name der Unit. Oder auf

eine Gruppe S1 bis S10 bezogen `count crew LKW1 == 11`. Es ergibt sich hierbei jedoch das Problem, dass bei Verlust der Gruppe der LKW nie abfährt. Der Vergleich `1 == 11` bezieht

sich auf einen Fahrer plus 10 Mann der Gruppe.

Bestimmte Einheit in Vehicle ansprechen

Um eine bestimmte Einheit in einem Panzer Auto Truck etc. anzusprechen wird der Name des Vehicle verwendet. Zu diesem Namen wird ein d für Driver ein g für gunner ein c für commander angehängt. Beispiel: `Name des Vehicle+d setdamage 1;`

Das sorgt dafür dass die Einheit intakt bleibt, der Fahrer jedoch tot ist. Beispiele.

`Name des Vehicle = car1`. Der Name des Driver ist `car1d` der gunner `car1g` und des Commanders `car1c`.

Bestimmte Einheit in Vehicle löschen

Um eine bestimmte Einheit in einem Vehicle zu löschen wie den Gunner oder Driver etc kann dieses ohne einen bestimmten zugeordneten Namen erfolgen. Dazu in die Initzeile der Einheit (Name des Vehicle tank) eintragen. Diese kleine Spielerei kann es dem Missionsbastler ermöglichen die Kampfkraft der Unit zu mindern. Ein Panzer ohne commander ist natürlich nicht so stark und hat ein MG weniger.

`deleteVehicle (gunner tank)`

`_unit=gunner tank; _unit setPos [0,0,0]; deleteVehicle _unit`

Einheit am Fallschirm

Um eine Gruppe zu Missionsbeginn an einem Fallschirm starten zu lassen können entweder Fallschirmtruppen gewählt werden oder wenn diese Truppenart nicht benötigt wird kann auch eine andere Einheit in einen Fallschirm gesetzt werden. Dazu erstellt im Editor einen

Fallschirm. Dieser sollte auf fliegen gestellt sein und über der Absprungzone in Position gebracht worden sein. Diesem Fallschirm gebt ihr einen Namen. (Fallschirm1,2,3,...). Jetzt

erstellt eure Soldaten. In der jeweiligen Initzeile wird jetzt der Befehl `this moveindriver`

`Fallschirm1` eingetragen. Dieses bewirkt das die unit in dem Fallschirm als driver platz

nimmt. So verfährt ihr mit eurer Gruppe. Name der Einheit oder this in der Initzeile

`moveindriver Name des Fallschirms`. Der Name der versetzten Einheit ist jetzt der des

Fallschirm+d. d steht für Driver. Beispiel Name der Einheit = Alex, Name des Fallschirm =

Air1, neuer Name der Einheit = Air1d

Nur bestimmter Typ darf die Einheit Nutzen

Wenn ein Vehicle von nur einer Art von Einheit genutzt werden darf, kann das über einen Auslöser gelöst werden.

Bedingung: `!isNull driver h_1`

Bei Aktivierung: `if(typeOf driver h_1 != "SoldierWPilot")then{ driver h_1 action["Eject",h_1]};`

Hierbei ist der Name der zu sperrenden Einheit h_1 und die Art der Unit die diese nur als driver nutzen darf ist SoldatWPilot

Damit können jetzt zum Beispiel Die Helikopter für Piloten nutzbar gemacht werden.

Während jede andere Einheit wieder aussteigt sobald sie als Driver in der Einheit sitzt.

Vehicle beschleunigen

Eine Erweiterung zum starten einer Gruppe in einem Helikopter ist den Helikopter zu Missionsbeginn eine Geschwindigkeit mitzugeben. Es ist ja etwas schade wenn eine Gruppe in einem fliegenden Helikopter sitzt und dieser zuerst von 0 beschleunigen muß. Dies verhindert man indem man diesem Vehicle einen Vector und einen Geschwindigkeitswert mitgibt. Dieser Vektor ist in die Richtungen x y und z aufgeteilt. Jedem dieser drei Richtungen können positive un9 negative werte zu geordnet werden.

Befehl: `Name der Einheit setvelocity [werx,werty,wertz]`

Vorsicht: Es gibt keine Grenzen bei diesem Befehl. Es ist möglich eine Einheit auf 4000 oder mehr zu beschleunigen. Um ein bereits in bewegtes Objekt weiter zu beschleunigen kann die mit der Addition von X Y Z gemacht werden. Es wird einfach die Bewegung ausgelesen und die Werte dazu addiert. Befehl:

`_object setVelocity [(velocity _object select 0) + x, (velocity _object select 1) + y, (velocity _object select 2) + z];`

x,y,z sind die Werte, bezogen auf die Welt.

Verhalten, Geschwindigkeit einer Gruppe

Jede Geschwindigkeit einer Einheit

Mit dem Befehl `setspeedmode` kann die Geschwindigkeit einer Einheit unabhängig der Einstellungen eines Wegpunktes verändert werden.

Zum Beispiel : `Name setspeedmode "Limited"`

"Limited"	langsame Geschwindigkeit
"Normal"	normale Geschwindigkeit
"Full"	maximale Geschwindigkeit

Verhalten einer Einheit

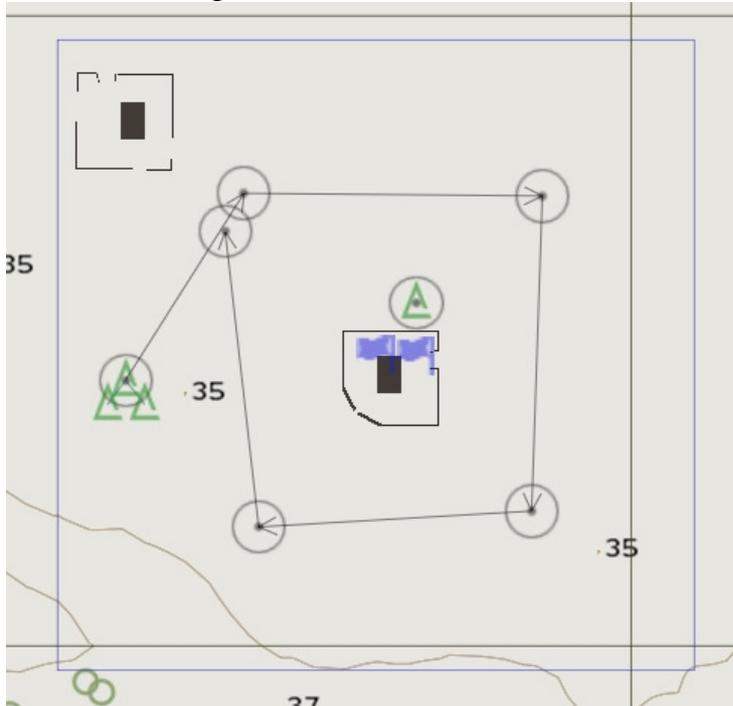
Mit dem Befehl `setbehavior` kann die Geschwindigkeit einer Einheit unabhängig der Einstellungen eines Wegpunktes verändert werden.

Zum Beispiel : `Name setbehavior "careless";`

"careless";	steht für Achtlos
"safe";	steht für Sicherheit
"aware";	steht für Wachsam
"combat";	steht für Kampf
"stealth";	steht für Tarnung

Wird also ein Wegpunkt einer Gruppe direkt auf die Anführer gesetzt in dem Geschwindigkeit begrenzt und Verhalten `save` gesetzt wird und diese eine Schleife ohne den Startwegpunkt läuft, kann zum Beispiel mittels Auslösers dies geändert werden. Dies bietet

sich an, wenn Alarm ausgelöst wird und alle Einheiten in einem Bereich auf Kampf und volle Geschwindigkeit gesetzt werden sollen. In diesem Fall haben wir eine Gruppe, die um eine Ölpumpe geht. Der ganze Bereich wird durch einen Auslöser abgedeckt, welcher Achse AB 100, Aktivierung Ost Einmal und den Namen Alarmzone hat.



sowie einen weiteren Auslöser mit Achse AB 0 und Aktivierung Alarm. Bei Aktivierung steht `{_x setBehaviour "aware"} forEach list Alarmzone; {_x setspeedmode "Normal"} forEach list Alarmzone;` (speedmode läuft unter beta 1.55 nicht)

Das führt dazu, dass alle Einheiten bei Alarm = true auf Wachsam umstellen und sich mit normaler Geschwindigkeit bewegen.

Feiger Feind

Es gibt die Möglichkeit einer AI einen Fluchtwert zu geben. Das bewirkt, dass bei starker Gegenwehr die Einheit oder die Gruppe sich zurückzieht und flieht. Es kann passieren, dass diese Einheiten sich neu formieren und erneut wiederkommen.

Befehl : `this allowFleeing 0.8`

Der Wert bestimmt den Punkt, wann das Fluchtverhalten beginnt. Ist dieser Wert 0, bleibt die Einheit und denkt noch nicht einmal an Flucht. 1 ist das maximale Fluchtverhalten.

Formation

Diese Befehle ändern die Formation der Gruppe.

Name des Gruppenleiters SetFormation "Line";

Es gibt die Formationen

Line	Die Gruppenmitglieder gehen in einer Reihe
Column	Die Gruppenmitglieder gehen im Gänsemarsch
StaggeredColumn	Die Gruppenmitglieder gehen im Gänsemarsch versetzt
Wedge	Die Gruppenmitglieder gehen in einem Keil
Echelon Left	Die Gruppenmitglieder gehen als Staffel ausgerichtet nach links
Echelon Right	Die Gruppenmitglieder gehen als Staffel ausgerichtet nach rechts
Vee	Die Gruppenmitglieder gehen nach einem V aufgereit.

Einheiten formieren

Um Einheiten zu einer Gruppe zu binden, bedient man sich dem Befehl:

`[name1,Name2] join group Name der Gruppe`

`[John,Tim] join group Sniper;`

In diesem Beispiel verlassen die beiden (John und Tim) ihre Gruppe und schließen sich Sniper an. Sollte keine Gruppe Sniper existieren wird diese erstellt und der Ranghöchste übernimmt die Führung. Wird eine Osteinheit einer Westgruppe zugewiesen, übernimmt diese den West Status. Es sei denn die Osteinheit ist der Rang Höchste. Dann ist die ganze Westgruppe plötzlich auf der Seiten von Ost.

Um eine Einheit einfach aus der Gruppe zu entfernen:

`[Name der Einheit] join grpnnull;`

Damit ist die Einheit aus der Gruppe entfernt und bleibt stehen. Einer Einheit (Jede Einheit ist automatisch eine Gruppe) kann jetzt ein Wegpunkt zugewiesen werden. Dies wird mit dem Befehl `group addWaypoint [center, radius]` realisiert.

Beispiel:

`wp1 = Name der Gruppe / Einheit addWaypoint [position player, 0];`

Was dazu führen würde das ein Wegpunkt Name wp1 auf der Position des Spielers erstellt würde, welcher den Radius 0 hat.

Waffen an die AI verteilen

Mit dem Befehl `Name der Einheit SelectWeapon " M136"` wird einem AT Soldaten der Befehl gegeben, die Panzerabwehr-Waffe zu wählen. Voraussetzung ist, dass die Waffe auch bei der Einheit vorhanden ist.

Waffen einer Einheit ändert man dadurch, dass ihr neue zugewiesen werden. Dies geht am besten in der Init Zeile einer Einheit. Zuerst ist es ratsam diese zu entwaffnen. Somit sind die Waffenslots frei und diese Einheit kann neu bewaffnet werden.

Um eine Einheit ganz zu entwaffnen. Befehl:

`removeAllWeapons this`

Natürlich kann dies auch über ein Skript oder Auslöser erfolgen. Es sieht jedoch etwas komisch aus wenn im Verlaufe einer Mission eine Einheit plötzlich ohne Waffe da steht.

Um dieser jetzt wieder neue zu zuweisen verwendet man den

Befehl: `Name der Einheit addWeapon "Typ der Waffe"`

Befehl: `Name der Einheit addmagazine "Typ des Magazins"`

Beispiel:

```
Removeallweapons this;
this addmagazine "30Rnd_9x19_MP5";
this addweapon " MP5A5 ";
this addmagazine "SmokeShellRed";
this addweapon "NVGoggles";
```

Es fällt auf das dieser Einheit, dessen Init Zeile hier ausgelesen wurde, zuerst die Magazine gegeben wurden und dann die Waffen. Dies hängt damit zusammen, dass die Einheit gleich eine geladene Waffe haben soll und nicht erst nachladen muss. Zur Verdeutlichung. Zuerst wurden alle Waffen entnommen,

1 MP5 Magazin,

1 Mp5,

1 rote Rauchgranate sowie ein Nachtsichtgerät eingefügt.

Der nächste Schritt ist jetzt Waffen und Munition in Vehicle und Kisten zu laden. Im Grunde funktioniert dies genau wie einer Einheit Waffen zu geben.

`clearWeaponCargo this;`

löscht alle Waffen in der Kiste

`clearMagazineCargo this;`

löscht alle Munition in der Kiste

```
this addweaponcargo ["M4A1",4];
this addmagazinecargo ["30Rnd_556x45_Stanag",6];
```

Packt 4 M4A1 ein.
Packt 6 Magazine ein.

Vehicle bewaffnen

Genau wie ein Soldat kann auch ein Vehikel um bewaffnet werden. Dies geschieht einfach durch das vergeben von Magazine und Launcher. Damit ist es Möglich der KA 50 Hellfire Raketen zu geben oder der Cobra AIM9 Raketen für Luftziele mitzugeben. Diese Raketen werden jedoch nicht dargestellt. Und es ist nicht möglich einem Helikopter Stinger Raketen zu geben oder etwas anderes aus dem Personen Sektor. Diese Befehle werden entweder in der Init einer Einheit oder in einem Skript Auslöser eingefügt. Beispiel:

```
RLuft1 addmagazine "8Rnd_Hellfire";
RLuft1 addweapon "HellfireLauncher";
RLuft1 addmagazine "4Rnd_Sidewinder_AV8B";
RLuft1 addweapon "SidewinderLaucher";
```

In diesem Fall wurde der Einheit RLuft1 eine Hellfire Rakete + Launcher und

eine Sidewinder Rakete + Launcher gegeben. Es ist wichtig, dass die Einheit auch den Launcher zu den Magazinen bekommt da sonst die Waffe (Munitionslot) nicht abgefeuert werden kann.

Prominente Einheiten

Jede Einheit hat von einer anderen einen Bekanntheitswert. Der **knowsAbout**-Wert. Über diesen wird das Verhalten der AI geregelt.

Beispiel: `_weise = Einheita knowsAbout Einheitb`

Mit diesem Wert regelt das Spiel jede Aktion der Einheiten untereinander.

Alarm abspielen

In vielen Missionen ist der Alarm ein zentraler Punkt. Eine Base wird angegriffen, der Spieler entdeckt und ein lautes Signal übertönt alles. Aber was ist das eigentlich, dieser Alarm? Der Alarm ist nichts anderes als eine der vielen in Arma eingebundenen Sounddateien. Diese können unter Effekte Play Alarm ausgelöst werden und sind frei abrufbar. Der Sound hat jedoch keine Auswirkung auf die AI Einheiten. Es empfiehlt sich also die Einheiten in dem Bereich zusätzlich auf den Verhaltensmodus Kampf zu stellen. Dies kann durch einen Auslöser mit Aktivierung (Seite wählen) einmal und als Zusatzbedingung eine Variable erfolgen und dann unter Aktivierung (Kampf) aufgerufen werden.

<p>Der Auslöser wird in diesem Beispiel über Funk Alpha angesprochen. Die Achse ist AB 0 / also global, einmalige Aktivierung. Unter Effekte wurden jetzt unter Stimme Alarm abspielen ausgewählt. Das führt dazu, das jetzt der Sound abgespielt wird sobald Radio alpha gewählt wird. Der Alarm kann natürlich auch anderes aktiviert werden. zB mit einer Variable oder einem Ereignis.</p>	<p>Einstellungen Auslöser:</p>
--	--------------------------------

<table border="1"> <tr><td>true</td></tr> <tr><td>Anonym:</td><td>Kein Sound</td></tr> <tr><td>Stimme:</td><td>Alarm abspielen</td></tr> <tr><td>Umgebung:</td><td>Kein Sound</td></tr> <tr><td>Auslöser:</td><td>Kein Sound</td></tr> <tr><td>Spur:</td><td>Keine Musik</td></tr> <tr><td>Typ:</td><td>KEINEN</td></tr> <tr><td>Effekt:</td><td>EINFACH</td></tr> </table>	true	Anonym:	Kein Sound	Stimme:	Alarm abspielen	Umgebung:	Kein Sound	Auslöser:	Kein Sound	Spur:	Keine Musik	Typ:	KEINEN	Effekt:	EINFACH	<table border="1"> <tr><td>Achse A</td><td>0</td><td>Achse B</td><td>0</td></tr> <tr><td>Winkel</td><td>0</td><td>Ellipse</td><td>Rechteck</td></tr> <tr><td>Aktivierung</td><td>Funk: Alpha</td><td>Einmal</td><td>Mehrfach</td></tr> <tr><td colspan="2">Vorhanden</td><td colspan="2">Nicht vorhanden</td></tr> <tr><td colspan="2">Von Westen entdeckt</td><td colspan="2">Von Osten entdeckt</td></tr> <tr><td colspan="2">Von Widerstand entdeckt</td><td colspan="2">Von den Zivilisten entdeckt</td></tr> <tr><td>Countdown</td><td>Min.: 0</td><td>Max.: 0</td><td>Mid.: 0</td></tr> <tr><td>Timeout</td><td></td><td></td><td></td></tr> <tr><td>Typ</td><td colspan="3">Keine</td></tr> <tr><td>Name:</td><td></td><td>Text</td><td></td></tr> <tr><td>Bedingung</td><td colspan="3">this</td></tr> <tr><td>Bei Akt.</td><td colspan="3"></td></tr> <tr><td>Bei Deak.</td><td colspan="3"></td></tr> <tr><td>Effekte</td><td>OK</td><td colspan="2">Abbrechen</td></tr> </table>	Achse A	0	Achse B	0	Winkel	0	Ellipse	Rechteck	Aktivierung	Funk: Alpha	Einmal	Mehrfach	Vorhanden		Nicht vorhanden		Von Westen entdeckt		Von Osten entdeckt		Von Widerstand entdeckt		Von den Zivilisten entdeckt		Countdown	Min.: 0	Max.: 0	Mid.: 0	Timeout				Typ	Keine			Name:		Text		Bedingung	this			Bei Akt.				Bei Deak.				Effekte	OK	Abbrechen	
true																																																																								
Anonym:	Kein Sound																																																																							
Stimme:	Alarm abspielen																																																																							
Umgebung:	Kein Sound																																																																							
Auslöser:	Kein Sound																																																																							
Spur:	Keine Musik																																																																							
Typ:	KEINEN																																																																							
Effekt:	EINFACH																																																																							
Achse A	0	Achse B	0																																																																					
Winkel	0	Ellipse	Rechteck																																																																					
Aktivierung	Funk: Alpha	Einmal	Mehrfach																																																																					
Vorhanden		Nicht vorhanden																																																																						
Von Westen entdeckt		Von Osten entdeckt																																																																						
Von Widerstand entdeckt		Von den Zivilisten entdeckt																																																																						
Countdown	Min.: 0	Max.: 0	Mid.: 0																																																																					
Timeout																																																																								
Typ	Keine																																																																							
Name:		Text																																																																						
Bedingung	this																																																																							
Bei Akt.																																																																								
Bei Deak.																																																																								
Effekte	OK	Abbrechen																																																																						

Alarm auslösen

Alarm soll ausgelöst werden sobald die Wache (Name: Wache1) den Spieler entdeckt und einen Schuss in den Himmel abgefeuert hat. Die Wache soll jedoch noch 2 Sekunden warten bevor diese feuert, damit der Spieler noch Zeit hat zu reagieren.

Erstellt eine Einheit Seite Ost Name Wache1. Bewaffnet diese in der ihr in der Init Zeile Waffentext einfügt. Da zu fügt ihr, [Wache1] exec "posten.sqs"

Quellcode: posten.sqs

```
_wache = this select 0;
@_wache knowsabout player > 0.4
~2
?not alive Wache1 goto: "exit"
Wache1 fire "AK47"
exit
```

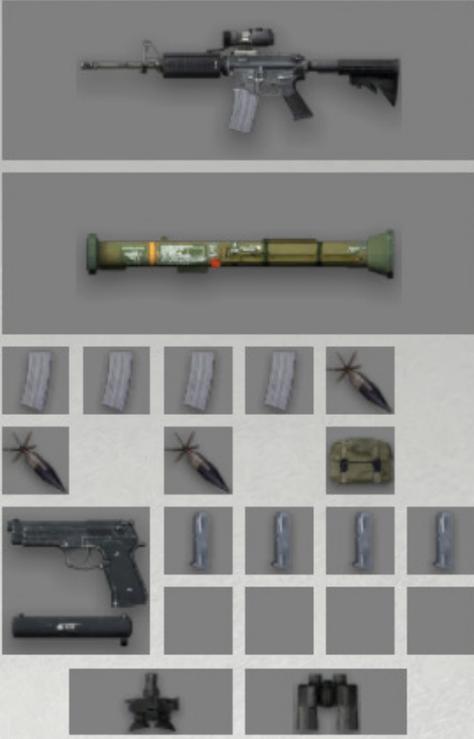
Jetzt hat der player 2 Sekunde Zeit die Wache aus zu schalten bevor diese einen Schuss abgibt. Es ist darauf zu achten das die Wache auch eine AK74 als Waffe hat. Ansonsten wird kein Schuss abgegeben.

Und für die Abfrage ob die Wache gefeuert hat. Wird in die Initzeile der Wache folgendes schreiben:

```
Wache1 addeventhandler ["fired", {alarm=true}]
```

Die Variable Alarm wird auf true gesetzt sobald die Wache geschossen hat. Diese kann dann weitere Aktionen wie Alarm Abspielen etc auslösen.

Waffen & Munition der Einheit erfassen

<p>Waffen ["M4AIM", "M136", "NVGoggles", "M9SD", "Binocular"] Magazin ["30Rnd_556x45_Stanag", "30Rnd_556x45_Stanag", "30Rnd_556x45_Stanag", "30Rnd_556x45_Stanag", "M136", "M136", "M136", "PipeBomb", "15Rnd_9x19_M9SD", "15Rnd_9x19_M9SD", "15Rnd_9x19_M9SD", "15Rnd_9x19_M9SD"]</p>  <p>Ausrüstung der Einheit:</p>  <p>Rang: Fahrzeug:</p>	<p>Um die Waffe der Einheit aus zu geben kann das als Textmitteilung geschehen. hint format ["%1", weapons Name der Einheit]; Beispiel: John ist der Name der Einheit. hint format ["%1", weapons John]; liefert die Namen der Waffen der Einheit. Dieses erfolgt als Array, ["Primere Waffe Gewehr", "Raketenwerfer", "Sekundäre Waffe Pistole", "Fernglas, Nachtsichtgerät"] Die Rein folge ist jedoch frei. Die Ausgabe erfolgt in der Rein folge in der die Waffen aufgenommen werden.</p> <p>Dasselbe kann mit den Magazinen gemacht werden. Beispiel: John ist der Name der Einheit. hint format ["%1", magazines John]; liefert die Namen der Magazine der Einheit. Dieses erfolgt als Array. Im linken Bild ist oben der Ausdruck zu sehen und unten das dazu gehörige Ladeverzeichnis von John.</p> <p>Dieses Array kann wie jedes andere auch ausgelesen werden. Position 1 mit <code>_this select 0</code>; usw.</p>
--	---

Tiefflug

Um ein Luftobjekt nicht auf der Standardhöhe fliegen zu lassen, besteht die Möglichkeit, der Einheit den Befehl **Name der Einheit flyinheight Angabe in Metern** zu geben.

Beispiel : **Luft1 flyinheight 20**

Luft1 ist der Name der Einheit. Dies geht natürlich auch nach oben. Wobei man darauf achten sollte, dass der Wegpunkt evtl. nicht in jeder Flughöhe zu erreichen ist. Achtet weitergehend darauf, dass die Strecken, die geflogen werden, keine Bäume haben, da ein Tiefflug von ca 5 Metern sonst in der nächsten Tanne endet.

Einheiten zu einer anderen Einheit gehen lassen

Es gibt weitere Wege, eine Einheit dazu zu bringen, einen Punkt auf der Karte anzusteuern.

Name der Einheit DoMove GetPos Zielort.

Befehl : **Spiderman DoMove GetPos Johan**

Spiderman ist der Name der Einheit, die zu der Einheit Johan geht.

Oder zu einem Objekt

Ersetzt man Johan durch ein Objekt, ObjectID(23542) und will jetzt Johan dahin befehligen.

Steht man vor dem Problem das man eine Game Logik setzen muss, welche mit (getPos

gameLogic nearestObject 23542) die Position liefert. Da das etwas sehr aufwendig ist kann

der Spiler auch gleich eine Logik oder einen Auslöser (Name Marke) setzen und mit dem

Befehl: **Name1 DoMove GetPos Name2** Achtung: Wenn ein Marker verwendet wird, wird die Position mit getMarkerPos ausgelesen.

Beispiel: **Spiderman DoMove GetMarkerPos Marker**

Der Feind in meinem Bett

Um eine Einheit (Gefangener) in einen Bereich mit Feinden zu stellen, und sicherzustellen, dass dieser nicht gleich über den Haufen geschossen wird, einfach in seine Init Zeile

this Setcaptive true eintragen. Das ganze kann natürlich auch über ein Skript erfolgen. (**Name Setcaptive true**). Um das rückgängig zu machen, wird **true** durch **false** ersetzt

Feindlicher Kamerad

Wenn sich jemand dem ArmA untypischen Ziel zuwendet eine MP Mission zu bauen welche auf ein DM Spiel hinausläuft kann dies dadurch erzielt werden das alle Freunde untereinander verfeindet werden. **Seitea setFriend [Seitea, 0]**; Als Seitea kann **WEST, EAST, Resistance** oder **Civilian** eingetragen werden. Danach sollte man sich vor seinen Kameraden in Acht nehmen. Alle Werte kleiner 0.6 sind feindlich eingestuft.

Gegner freundlich

Um einen Gegner Freundlich zu stimmen kann jede Einheit mit **this Setcaptive true** angesprochen werden. Dies ist jedoch eine unnötige Aufwendige Methode. Das Selbe ist leichter dadurch zu erreichen, das einfach die ganze Seite freundlich gestimmt wird. Quell Code:

WEST setFriend [EAST, 0.7];

EAST setFriend [WEST, 0.7];

Die Aufrufe müssen von jeder Seite zu jeder Seite ausgeführt werden. Da ansonsten die andere Seite immer noch die andere bekämpft.

Gegner wieder feindlich

Um das ganze wieder umzukehren wird einfach in einem Skript Auslöser geschrieben.

WEST setFriend [EAST, 0.1];

EAST setFriend [WEST, 0.1];

Und schon fangen beide Seiten an aufeinander zu schissen. Mit dieser Methode wird es dann Möglich auch die Zivilisten einer Seite angehörig zu machen.

Suchscheinwerfer ein und aus schalten

Die Suchscheinwerfer erfüllen jede Mission mit leben. Was ist aber wenn ich möchte dass ein Scheinwerfer ohne Besatzung leuchtet. Im Prinzip ist das nicht möglich. Jedoch eine Spiellogik kann genau wie ein Soldat zum einsteigen oder aussteigen genutzt werden. Hierzu einfach eine Logik erstellen und in die Init Zeile dieser schreiben:

this moveincargo Name des Scheinwerfers

Damit erreicht man dass dieser Scheinwerfer leuchtet.

Spielbare Einheit hinzufügen

Switchbare Einheit hinzufügen. Es besteht die Möglichkeit, eine spielbare Einheit in das Spiel hinzuzufügen. Dazu wird die Einheit im Editor als Spielbar deklariert oder zu einem späteren Zeitpunkt hinzugefügt. Dies geht mit dem Befehl : **AddSwitchableUnit Name der Einheit.**

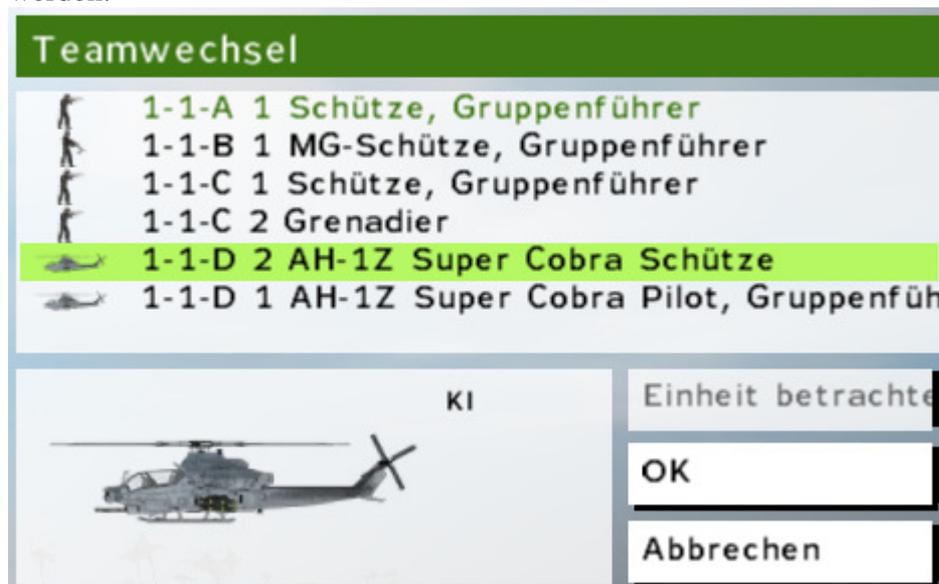
Beispiel : **AddSwitchableUnit Johan**

Man kann auch eine steuerbare Einheit mit dem Befehl : **RemoveSwitchableUnit Name der Einheit** streichen.

Beispiel : **RemoveSwitchableUnit Johan**

Johan ist der Name der Einheit. Dier Name wurde einer Einheit gegeben die im Editor platziert wurde.

Bei normaler Tastaturbelegung kann durch drücken der „t“ Taste diese Menue aufgerufen werden. In diesem Menue sind alle spielbaren Einheiten aufgelistet, und können gewählt werden.



Der MissionsOrdner

Der Missionsordner wird beim speichern automatisch in dem aktiven Benutzerprofil erstellt. Diesen findet ihr unter:

Angemeldet mit dem PC Namen

C:\Dokumente und Einstellungen\Alex.Sworn\Eigene Dateien\Arma\missions

Angemeldet mit einem unabhängigen Namen

C:\Dokumente und Einstellungen\Drake Starkiller\Eigene Dateien\Arma Other Profiles\D%2eStarkiller\missions



Mission.sqm In diesem Ordner ist nach dem ersten Speichern die Mission.sqm. In dieser ist das

Abbild vom Editor als Quelltext gespeichert.



fallschirm.sqs Wenn jetzt im Editor über einen Auslöser oder durch eine Init Zeile ein Skript angefordert wird muß dieses zuerst im Ordner angelegt werden.



briefing.html allgemeins Briefing



briefing.German.html deutsches Briefing

Weitergehend ist hier das Briefing zu finden. In diesem Briefing sind Missionsziele beschrieben und Einsatzziele bestimmt. Es ist in der Mission im Pfad auf der Karte zu sehen. Die Briefings werden in html geschrieben.



overview.html allgemeines Overview



briefing.German.html deutsches Overview

Singleplayer-Einsätze werden in der Auswahl von einem kleinen Bild und einer kleinen Nachricht begleitet. Die Overview werden in html geschrieben.



Library.paa Das Bild für die Mission Briefing oder Overview wird hier auch eingefügt, oder Bilder für Fahnen und so weiter.

Weitere Typen sind die stringtable.csv, in der die Texte für Funkmitteilungen definiert werden können damit die Mitteilung in der richtigen Sprache erscheint (siehe stringtable.csv). Sowie Funktionen und eine Reihe von weiteren Daten. Weitergehend besteht die Möglichkeit, einige Ordner zu erstellen. Diese haben keine Bedeutung und dienen nur der Übersicht.

Missions.sqm

Die Missions.sqm ist die eigentliche Mission. In dieser Datei sind alle im Editor platzierten Einheiten gespeichert. In der Datei finden sich die Haupteinstellungen für verwendete Addon Wetter Start, Wetter Ende, Tageszeit, Datum wieder.

Wie bekomme ich die Positionsdaten eines Objektes um damit zu arbeiten?
(eine Kameraeinstellung, Position an der eine Rauchgranate oder Flare erscheinen soll...)

1. Erstellt im Editor an der gewünschten Position ein Objekt oder Einheit.
2. Benennt diese Einheit um sie später in der Missions.sqm zu finden (z.B: Peter)
3. Nun geht in den Spieleordner eurer Mission. Öffnet die Missions.sqm mit dem Editor. Dieses Programm ist bei Windows unter Programme – Zubehör – Editor zu finden.
4. Ist die Missions.sqm mit dem Editor geöffnet, könnt ihr nun die Suchfunktion des Editors nutzen. Dazu im Editor – Bearbeiten – Suchen aktivieren. Gebt den Namen eurer Einheit ein (Peter). Das Programm springt zu dem Eintrag Peter.

```
class Item0
{
    position[]={ 16624.314453,146.800522,14429.284180};
    id=0;
    side="WEST";
    vehicle="TeamLeaderW";
    player="PLAY CDG";
    leader=1;
    rank="SERGEANT";
    skill=1.000000;
    text="Peter";
    init=" ";
};
```

5. Nun seht ihr die Positionsdaten. Diese einfach kopieren und dann in eurem Skript oder init Eintrag mit Strg+V einfügen.
6. ACHTUNG: Um mit diesen Daten arbeiten zu können ist es notwendig die mittlere Zahl zu löschen und die Höhen Angabe als dritten Wert anzuschliessen. Die Eingabe erfolgt in der mission.sqm [X,Z,Y] und für Funktionen wie setPos etc. ist es notwendig diese auf [X,Y,Z] umzustellen.

- a) { 16624.314453,**146.800522**,14429.284180};
- b) { 16624.314453, ,14429.284180};
- c) { 16624.314453,14429.284180,**10**};

Hier wurde der mittlere Eintrag gelöscht und die Höhe 10m angefügt. Das zu erstellende Objekt wird also in 10m Höhe erzeugt.
Die erstellte Einheit oder Objekt mit dem Namen Peter kann danach gelöscht werden.

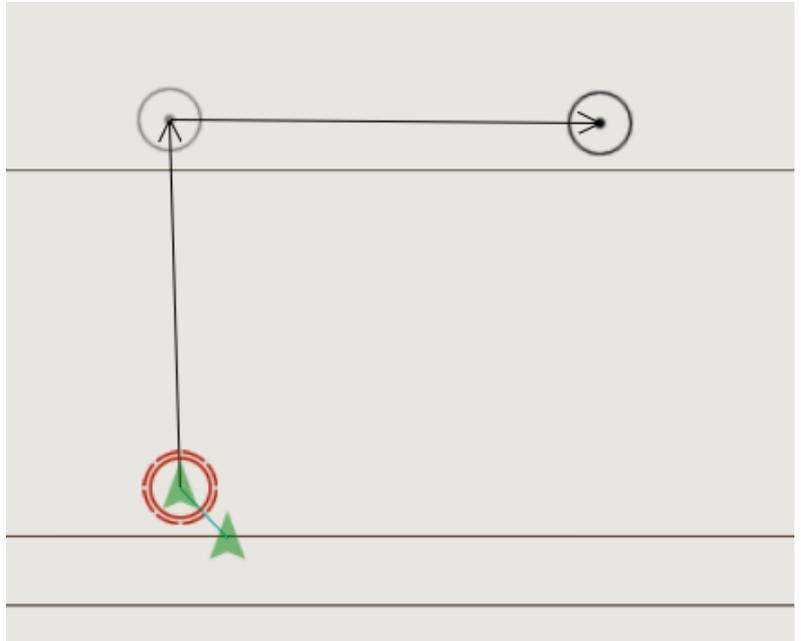
Quellcode Mission.sqs:

Editor Beispiel 1 Gruppe zwei Einheiten, zwei Wegmarken:

```

version=11;
class Mission
{
    addOns[]=
    {
        "cacharacters",
        "sara"
    };
    addOnsAuto[]=
    {
        "cacharacters",
        "sara"
    };
    randomSeed=1600003;
    class Intel
    {
        startWeather=0.100000;
        forecastWeather=0.300000;
        year=2007;
        month=6;
        day=7;
        hour=8;
    };
    class Groups
    {
        items=1;
        class Item0
        {
            side="WEST";
            class Vehicles
            {
                items=2;
                class Item0
                {
                    position[]={9661.832031,139.994995,10034.041992};
                    id=0;
                    side="WEST";
                    vehicle="SoldierWB";
                    player="PLAYER COMMANDER";
                    leader=1;
                    skill=0.600000;
                };
                class Item1
                {
                    position[]={9668.633789,139.994995,10026.719727};
                    id=1;
                    side="WEST";
                    vehicle="SoldierWB";
                    skill=0.600000;
                };
            };
        };
        class Waypoints
        {
            items=2;
            class Item0
            {

```



```

        position[]={9660.311523,139.994995,10087.313477};
        class Effects
        {
        };
        showWP="NEVER";
    };
    class Item1
    {
        position[]={9722.205078,139.994995,10086.795898};
        class Effects
        {
        };
        showWP="NEVER";
    };
};
};
};
class Intro
{
    addOns[]=
    {
        "sara"
    };
    addOnsAuto[]=
    {
        "sara"
    };
    randomSeed=2675715;
    class Intel
    {
        startWeather=0.100000;
        forecastWeather=0.300000;
        year=2007;
        month=6;
        day=7;
        hour=8;
    };
};
class OutroWin
{
    addOns[]=
    {
        "sara"
    };
    addOnsAuto[]=
    {
        "sara"
    };
    randomSeed=685571;
    class Intel
    {
        startWeather=0.100000;
        forecastWeather=0.300000;
        year=2007;
        month=6;
        day=7;
        hour=8;
    };
};
class OutroLoose
{

```

```

addOns[]=
{
    "sara"
};
addOnsAuto[]=
{
    "sara"
};
randomSeed=6011395;
class Intel
{
    startWeather=0.100000;
    forecastWeather=0.300000;
    year=2007;
    month=6;
    day=7;
    hour=8;
};
};

```

hier werden die verwendeten Addons aufgelistet. Dabei ist zu berücksichtigen, dass alle beim editieren verwendeten Einheiten hier aufgelistet werden. Daher kann es zu Fehlermeldungen kommen das Addons nicht verfügbar sind, selbst wenn mit diesen Addons nicht editiert wurde. Abhilfe kann geschaffen werden, in dem diese Einträge vor dem exportieren zu MP Missionen von Hand in der Missions.sqs gelöscht werden.

Hier werden die Missionsdaten Zeit, Wetter Start, Wette Ende und Datum eingetragen und können hier nach Belieben verändert werden.

Unter class Group sind alle Einheiten aufgeführt.

Hier ist die Spielereinheit aufgeführt, (Typ, Position, Fähigkeiten, Gesundheit, Einträge in dessen Init, der Name etc) und die Einheiten der Gruppe vereinbart. Die Anzahl der Gruppen sowie die Anzahl der Gruppenmitglieder sind unter items angegeben. Er wird ab 0 aufwärts gezählt.

class Waypoints enthält die Daten zu den Wegmarken. (Typ, Order, Position Radius der Platzierung, etc)

class intro enthält die Daten zum Intro. Hier sind dann wieder Einheiten Wegmarken etc aufgeführt. (nur wenn der Missionsbauer eins erstellt hat.)

class OutroWin enthält die Daten zum Abspann gewonnen. Hier sind dann wieder Einheiten Wegmarken etc. aufgeführt. (nur wenn der Missionsbauer eins erstellt hat.)

class OutroLoose enthält die Daten zum Abspann verloren. Hier sind dann wieder Einheiten Wegmarken etc. aufgeführt. (nur wenn der Missionsbauer eins erstellt hat.)

Description.ext

Klassendeklarationen

Diese Deklarationen gehören in die Description.ext (.ext Abkürzung für Extension). Bei dem Thema ist es am Besten, wenn man sich einen vernünftigen Programmierstil aneignet.

Bei jedem ist der etwas anders, daher nehmen wir den, welcher Standard ist und in ähnlicher Form von den meisten so genutzt wird:

```
/*Deklaration von Klassen/Funktionen*/
/* Oberklasse. Vorgegeben von OFP ist eine Art Bibliothek, wo festgelegt ist, wie die sounds abgespielt werden und in welcher syntax die Deklaration erfolgen muss. kann man mit #include <library> in C vergleichen */
```

```
class Cfgsounds
{

    sounds[] = {BLW_tut1,BLW_tut2,BLW_tut3,BLW_tut4};

    /* Subklasse/n */
    class BLW_tut1 {

        name    ="";
        sound[] ={"eins.ogg", db-12, 1.0};
        titles[] = {0,"Text"};
    };
};
```

Im Grunde ist das Deklarieren in ArmA so, als ob man in C lauter viele kleine Funktionen schreibt, und diese in der Hauptfunktion main ausgibt.

Quellcode eines C programmes:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

void funk(void);
int main() {
    funk();
    return 0;
    getch();
}
void funk(){
    int koordinate //Deklaration in C. int=Integer und das heißt ganze zahl
                  In diese Variable können nur ganzzahlen geschrieben werden.
    printf("koordinaten einlesen");
    scanf("%i",&koordinate);
    printf("eingegebene Koordinate:%i",koordinate);
}
```

Identität festlegen

Es gibt die Möglichkeit einzelnen AIs eine feste Identität zu geben. Dies betrifft das Gesicht, die Stimme, Stimmlage und Brille. In der Initzeile einfügen: `this setIdentities " Alex"`

```
class CfgIdentities
{
    class Alex
    // Einbinden mit: this setIdentities "Alex"
    {
        name = "Alex";
        face="Face10";
        glasses="None";
        speaker="Dan";
        pitch=1.1;
    };
};
```

oder die andere Möglichkeit, im Editor mit `Name der Einheit SetFace "Face33"` das Gesicht direkt zu weisen.

Mission freischalten

Es gibt eine Möglichkeit eine Mission zu sperren und an andere zu koppeln. Dieses wird auch Keysystem genannt.

```
keys[] = {"key1","key2","key3"};    Hier werden die möglichen Keys aufgeführt
keysLimit = 2;                      Anzahl der benötigten Keys.
doneKeys[] = {"key4"};              Name des Key welchen der Spieler erhält.
```

Typ der Mission

Es macht Sinn den Typ der Mission zu bestimmen. Damit weiß der Spieler welche Art er von Mission auswählt. Dies ist eher bei MP Missionen wichtig.

```
class Header
{
    gameType = COOP;                Hier wird der Typ der Mission bestimmt
    minPlayers = 1;                 Anzahl der min Player
    maxPlayers = 10;                Anzahl der max Player
};
```

Es gibt folgende Missionstypen. Death Match, Capture the Flag, Flag Fight, Cooperative Mission, Team Mission, Sector Control und Hold Location

Missionsname

Hier wird bestimmt, welcher Name beim Laden einer Mission angezeigt wird. Dazu wird in die Description.ext folgendes als Beispiel geschrieben.

```
onLoadMission=Alex.Sworn Missionsbeispiel 4
```

Hier wird bestimmt, welcher Name beim Laden eines Missions Intro angezeigt wird.

```
onLoadMissionsIntro = Alex.Sworn Intro4
```

Zeitanzeige

Weitergehend besteht die Möglichkeit, dass die Zeit während des Ladens angezeigt wird. Dazu in der Description.ext folgendes eintragen.

`OnLoadIntroTime = False`
`onLoadMissionTime = False`

Ausrüstung für den Spieler

Dem Spieler stehen einige Geräte zur Verfügung. Unter anderem die Karte (m). Auf der Karte können nun verschiedene Geräte genutzt werden: GPS, Kompass, Funkgerät, Uhr sowie das Briefing und das Endbriefing nach Vollendung der Mission. Um diese Dinge in einer Mission zu nutzen oder nicht zu nutzen, kann in der description.ext jedes Gerät an oder ausgeschaltet werden.

Hierfür wird das Gerät mit dem Namen aufgeführt und mit 1 für da oder 0 für nicht vorhanden gekennzeichnet.

`ShowCompass = 0` : zeigt den Kompass nicht an
`ShowGPS=1` : zeigt das GPS an
`ShowWatch = 1` : zeigt die Uhr an
`ShowRadio = 1` : zeigt das Funkgerät an
`ShowMap = 1` : zeigt die Karte an
`ShowNotePad = 0` : zeigt das Briefing nicht an
`ShowDeBriefing = 0` : zeigt das Abschlussbriefing nicht an

Punkte für den Spieler

Im ArMA gibt es bei Missionen ein Punkte System. Dieses sorgt dafür, dass der Spieler nach Abschluss der Mission eine Bewertung erhält. Um diese zu regeln (was ist viel, was wenig), werden diese Werte in der Description.ext bestimmt.

`minScore=500`
`AvgScore =750`
`maxScore=1000`

Diese Punkte erreicht der Spieler durch Abschüsse, die er in der Mission erzielt hat.

Weitergehend besteht die Möglichkeit, einer Einheit direkt Punkte zu geben. Dies kann in Auslösern, Skripten, Wegpunkten erfolgen. Zum Beispiel `Alex addRating 200000`. Das würde der Einheit mit dem Namen Alex auf dessen momentanen Punktestand 200000 Punkte geben. Das geht natürlich auch in die andere Richtung. `Alex addRating -200000`. Das würde der Einheit mit dem Namen Alex auf dessen momentanen Punktestand -200000 Punkte geben.

Waffenwahl im Briefing

Um Waffen in der Mission vor Missionsbeginn zu wählen, ist es notwendig, diese in der description.ext zuvor zu deklarieren. Dies wird unter den Klassen class weapons und class magazines gemacht.

Hier werden alle Waffentypen bestimmt (Class Weapons)

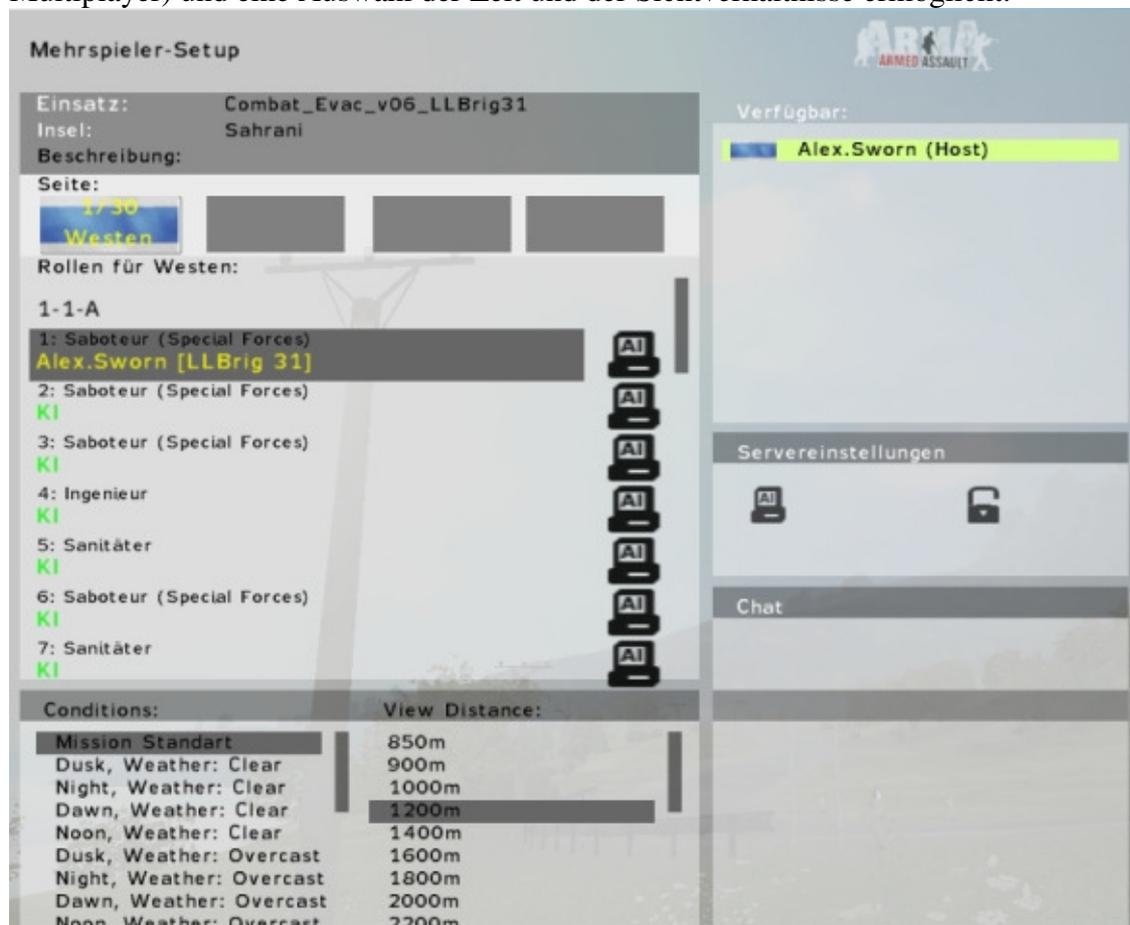
```
class Weapons
{
    class M4A1SD
    {
        count = 6;
    };
};
```

Hier werden alle Magazintypen bestimmt (Class Magazines)

```
class Magazines
{
    class 15Rnd_9x19_M9SD
    {
        count = 100;
    };
};
```

Beispiel einer description.ext einer Multiplayer Mission

Es werden hier neben Name beim Laden und Geräte für den Spieler auch der Respawn (siehe Multiplayer) und eine Auswahl der Zeit und der Sichtverhältnisse ermöglicht.



Im unteren Teil sind zwei Auswahlmöglichkeiten dazu gekommen. Eine ist für das Wetter und die Tageszeit, die andere ist die Sichtweite, die der Server bereitstellt. Weitergehend sollte in der init.sqs die entsprechenden Werte eingegeben sein.

```
; WETTER OPTIONEN
?(Param1 == 1) : {0 setFog 0.34; skiptime 0; 0 setOvercast 0.3;};
?(Param1 == 2) : {0 setFog 0; skiptime 23.20; 0 setOvercast 0;};
?(Param1 == 3) : {0 setFog 0; skiptime 4; 0 setOvercast 0;};
?(Param1 == 4) : {0 setFog 0; skiptime 9.20; 0 setOvercast 0;};
?(Param1 == 5) : {0 setFog 0; skiptime 15; 0 setOvercast 0;};
?(Param1 == 6) : {0 setFog 0; skiptime 23.20; 0 setOvercast 0.7;};
?(Param1 == 7) : {0 setFog 0; skiptime = 4; 0 setOvercast 0.7;};
?(Param1 == 8) : {0 setFog 0; skiptime 9.20; 0 setOvercast 0.7;};
?(Param1 == 9) : {0 setFog 0; skiptime 15; 0 setOvercast 0.7;};
?(Param1 == 10) : {0 setFog 0.4; skiptime 23.20; 0 setOvercast 1;};
?(Param1 == 11) : {0 setFog 0.4; skiptime 4; 0 setOvercast 1;};
?(Param1 == 12) : {0 setFog 0.4; skiptime 9.20; 0 setOvercast 1;};
?(Param1 == 13) : {0 setFog 0.4; skiptime 15; 0 setOvercast 1;};
?(Param1 == 14) : {0 setFog 0.7; skiptime 23.20; 0 setOvercast 0.7;};
?(Param1 == 15) : {0 setFog 0.7; skiptime 4; 0 setOvercast 0.7;};
?(Param1 == 16) : {0 setFog 0.7; skiptime 9.20; 0 setOvercast 0.7;};
?(Param1 == 17) : {0 setFog 0.7; skiptime 15; 0 setOvercast 0.7;};
?(Param1 == 18) : {0 setFog 0.97; skiptime 23.20; 0 setOvercast 0.7;};
?(Param1 == 19) : {0 setFog 0.97; skiptime 4; 0 setOvercast 0.7;};
?(Param1 == 20) : {0 setFog 0.97; skiptime 9.20; 0 setOvercast 0.7;};
?(Param1 == 21) : {0 setFog 0.97; skiptime 15; 0 setOvercast 0.7;};

; ZEIT OPTIONEN
if(!(param2 > 850)&&!(param2 < 10000));
setviewdistance param2;
setterraingrid 12.5;
```

Beispiel : description.ext

```
class Header
{
    gameType = Coop;
    minPlayers = 2;
    maxPlayers = 30;
};
OnLoadintroTime = true
OnLoadMissionTime = true
Onloadintro = Alex presents
onLoadMission = Ambush

minScore=500;
avgScore=2500
maxScore=3000

ShowGPS = 1
ShowCompass = 1
ShowRadio = 1
```

```
ShowMap = 1
ShowNotePad = 1
ShowWatch = 1
ShowDebriefing = 1
```

```
respawn = 3
respawnvehicle = 3
respawnVehicleDelay = 10
respawndelay = 6
disabledAI = 1
```

```
statsColumn=2;
titleParam1 = "Conditions:";
valuesParam1[] = { 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21 };
defValueParam1 = 1;
textsParam1[] = {"Mission Standart", "Dusk, Weather: Clear", "Night, Weather: Clear", "Dawn,
Weather: Clear", "Noon, Weather: Clear", "Dusk, Weather: Overcast", "Night, Weather:
Overcast", "Dawn, Weather: Overcast", "Noon, Weather: Overcast", "Dusk, Weather:
Storm", "Night, Weather: Storm", "Dawn, Weather: Storm", "Noon, Weather: Storm", "Dusk,
Weather: Mist", "Night, Weather: Mist", "Dawn, Weather: Mist", "Noon, Weather:
Mist", "Dusk, Weather: Fog", "Night, Weather: Fog", "Dawn, Weather: Fog", "Noon, Weather:
Fog"};
titleParam2 = "View Distance:";
valuesParam2[] =
{850,900,1000,1200,1400,1600,1800,2000,2200,2400,2600,2800,3000,3200,3400,3600,3800,
4000,6000,8000,10000};
textsParam2[] =
{"850m", "900m", "1000m", "1200m", "1400m", "1600m", "1800m", "2000m", "2200m", "2400m",
"2600m", "2800m", "3000m", "3200m", "3400m", "3600m", "3800m", "4000m", "6000m", "8000
m", "10000m"};
defValueParam2 = 1200;
```

```
class CfgMusic
{
tracks[] = { voodoo, };
class voodoo
{
name = " voodoo ";
sound[] = {\music\voodoo.ogg, db+25, 1.0};
};
};
```

Elegante Lösung

Es wird schnell deutlich dass es von Vorteil ist Themen auszulagern. Dazu werden im Missionsordner externe Dateien angelegt: Name der Datei.hpp. In diesen externen Dateien kann nun das Thema vereinbart werden. Dies ist bei Einbinden von Musikstücken oder Waffenwahl, oder Ausrüstung für den Spieler sehr hilfreich da nur die jeweiligen Informationen vorhanden sind. Dies erleichtert Änderungen. Beispiel: description.ext mit eingebundenen Externen Dateien.

```
onLoadMission=Testmission;
```

```
#include "gegen.hpp"  
#include "mp.hpp"
```

Das ist alles was in der description.ext steht. Es werden die Dateien gegen.hpp für die Ausrüstung,
Quellcode gegen.hpp:

```
ShowCompass = 1  
ShowGPS=1  
ShowWatch = 1  
ShowRadio = 1  
ShowMap = 1  
ShowNotePad = 1  
ShowDeBriefing = 1
```

sowie die Datei mp.hpp
Quellcode mp.hpp:

```
respawn = 3  
respawnvehicle = 3  
respawnVehicleDelay = 10  
respawndelay = 6  
disabledAI = 1
```

```
class Header  
{  
    gameType = Coop;  
    minPlayers = 2;  
    maxPlayers = 35;  
};
```

eingebunden. Dies kann natürlich mit beliebig vielen Dateien gemacht werden. (Sounds, Music, Waffenwahl... etc.) Dadurch wird eine description.ext nun sehr übersichtlich und einfach zu erweitern. Weitergehend können Teile einfach bearbeitet werden ohne Stunden lang Teile zu suchen. Vereinbarungen zum Einbinden.

Es wird dabei unterschieden zwischen #include "filename" und #include <filename>. "filename" wird für das Einbinden von Dateien, welche im selben Ortner sich befinden verwendet, <filename> startet ein Suche, welche nach implementierungsabhängigen Regeln ausgeführt wird. Der Übersetzer (Beim Starten der Mission) ersetzt jetzt den filename durch den Text in der Datei.

Textersatz

Eine Definition hat die Form #define NAME Ersatztext. Diese Möglichkeit gibt dem Programmierer die schnelle Methode viele Einträge in einem Text zusammen zu ändern. Der Übersetzer fügt bei allen definierten Namen jetzt den Text ein. So ist es Möglich zB bei vielen Einheiten die Waffen zu ändern, in dem #define Nachtsicht this addweapon "NVGoggles". Jetzt würde jeder Einheit ein NV hinzugefügt werden.

KI in der Abschussliste aufführen

hierzu wird in die description.ext der Eintrag

AIKills=1;

geschrieben. Jetzt wird jede Einheit aufgeführt und die Abschüsse angezeigt.

Briefing

Zu einer richtigen Mission gehört immer ein Briefing, sowie eine ausführliche Beschreibung der Ziele die erfüllt werden müssen. Quellcode:

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=windows-1250">
<meta name="GENERATOR" content="vb">
<title>Name des Briefings</title></head>
<body bgcolor="#FFFFFF">
<h2>
<a name="Main"></a>
</h2>
<!Hier werden die Notizen reingeschrieben>
<h6>Ueberschrift<br><br>
NOTIZEN
<br></h6>
<!Ende der Notizen>
<hr>
<!Hier wird der Missionsplan eingefügt>
<p><a name="Plan"></a>
<p>.
Missionsplan
</p>
<hr>
<!Missionsziele>
<p><a name="Obj_1"></a>Hier steht die Beschreibung von Ziel 1. </a></p>
<hr>
<p><a name="Obj_2"></a>Hier steht die Beschreibung von Ziel 2. <a
href="marker:NamedesMarker">Kartenmarkierung</a> </p>
<hr>
<p><a name="Obj_3"></a>Hier steht die Beschreibung von Ziel3. </p>
<hr>
<!Ende des Missionsplans>
<!Debriefing>
<hr><BR>
<h2><a name="Debriefing:End1">Ueberschrift Ende1</a></h2><BR>
```

```

<p>Ende1 Text</p><BR><hr><BR>
<h2><a name="Debriefing:End2">Ueberschrift Ende2</a></h2><BR>
<p>Ende2 Text</p><BR><hr><BR>
<!Ende des Debriefings>
</body>
</html>

```

Briefing

Notizen:

Überschrift	Diese Überschrift wird über den Notizen angezeigt.
NOTIZEN	Bei NOTIZEN wird der Text der Notizen eingegeben.

Missionsplan:

Missionsplan	Hier wird der Text der Notizen hingeschrieben.

Missionsziele:

Ziel 1	Hier wird der Text des ersten Ziels hingeschrieben
Ziel 2	Hier wird der Text des ersten Ziels hingeschrieben
Ziel 3	Hier wird der Text des ersten Ziels hingeschrieben

Verschiedene Befehle

 	Steht für einen Zeilenumbruch
<p> </p>	<p> steht für den Beginn eines Paragraphen und </p> für das Ende des Paragraphen.
<hr>	Trennung der Seite

 Kartenmarkierung

Stellt einen Link zu einer Kartenmarkierung her, sodass die Karte durch einen Klick auf den Link an die Position des Markers bewegt wird. NamedesMarker ist durch den entsprechenden Markernamen zu ersetzen.

Hinzufügen weiterer Missionsziele:

Die Anzahl der Missionsziele kann beliebig erweitert werden. Dies geschieht durch Hinzufügen von: <p>, wobei n für die Nummer des Ziels steht.

Beispiel:

<p>Hier steht die Beschreibung von Ziel 4</p>

Hinzufügen weiterer Seiten

Um eigene Seiten einem Briefing hinzu zufügen müssen diese als separate Einträge im HTML Quellcode. Dazu braucht man erstmal einen Link auf der Hauptseite, der so aussieht:

WasderSpielerlesenkann

Die HTML Hauptseite

Um die Seite überhaupt zu erstellen, muss man sie von der Hauptseite trennen. Dies geschieht durch den Befehl:

```
<hr>
```

Name des HTML Briefings:

```
<a name="NamederSeite"></a>
```

Debriefing:

Ueberschrift Ende1	Hier wird die Überschrift für das Debriefing Ende 1 festgelegt
Ende1 Text	Hier wird der Text für das Debriefing Ende 1 reingeschrieben

Missionsziele abhaken:

Um Missionsziele abzuhaken, wird der Befehl: "n" **ObjStatus** "done" verwendet, wobei n für die Nummer des Ziels steht.

Dieser Befehl kann z.B. in einem Auslöser verwendet werden.

Beispiel:

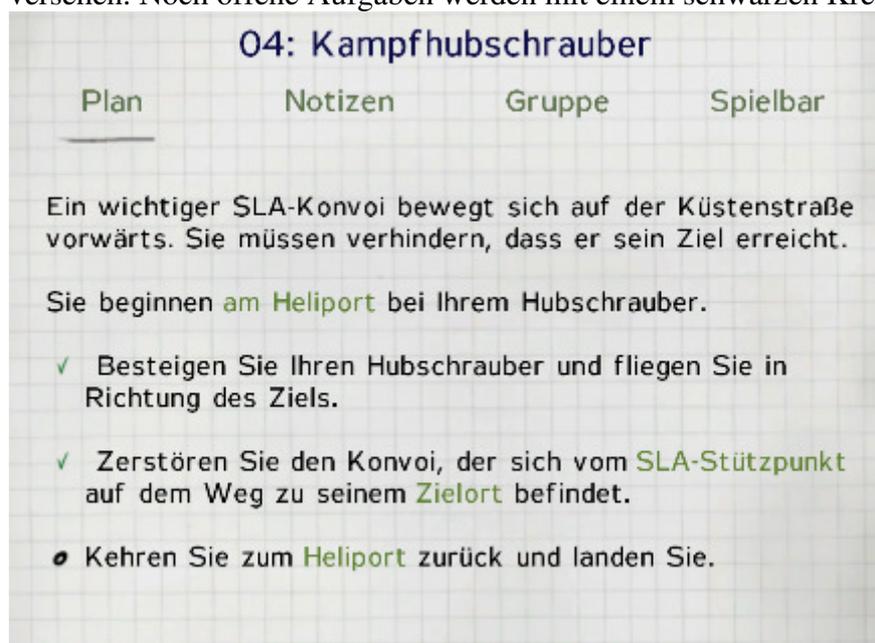
```
"1" ObjStatus "done"      weist Obj_1 als erfolgreich ab.
"2" ObjStatus "failed"    weist Obj_2 als nicht erfolgreich aus
"3" ObjStatus "hidden"    weist Obj_3 als nicht sichtbar für den Spieler aus
"4" ObjStatus "visible"   weist Obj_4 als sichtbar für den Spieler aus
```

Um den Spieler darauf aufmerksam zu machen, dass das Ziel erfüllt wurde, kann man noch mit dem hint Befehl eine entsprechende Meldung ausgeben.

Beispiel:

```
hint "Ziel 1 erfüllt"
```

Erledigte Aufgaben werden grün abgehakt, gescheiterte werden mit einem roten Kreuz versehen. Noch offene Aufgaben werden mit einem schwarzen Kreis gekennzeichnet.



Die Overview

Die Overview gibt im Singleplayer Spiel eine kurze Information über die auswählbare Mission.

Bei Multiplayer Missionen gibt es diese Option nicht. Die Overview ist genau wie das Briefing eine .html Datei.

Ein Overview sieht so aus:

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=windows-1250">
<meta name="GENERATOR" content="VB">
<title>Overview</title>
</head>
<body bgcolor="#FFFFFF">
<br>
<h2 align="center"><a name="Main">Kapitel 1</a></h2>
<p align="center"></p>
<br>
<p>
An Armed Assault Mission<br>
<br>
</p>
</body>
</html>
```

Zu sehen sind dann im Endeffekt nur die Zeilen 9 bis 15. Nun werden wir das mal auseinander nehmen.

Auf Zeile 10 wird ein Titel eingetragen der in großer Schrift erscheinen wird (Achtung, dies ist nicht der Missionsname). Darauf folgt ein Bild. Der Name des Bildes ist Morgensonne_256.jpg. Das Bild muss die Maße 2^n erfüllen also 32, 64, 128, 256, 512. Die Größe in der man es sehen wird, ist danach definiert. Und zwar wird es mit 64 Pixel Höhe und mit 64 Pixel Breite angezeigt. Danach kann man ein wenig Text finden. Das Overview lässt sich, wenn es denn gefixt würde, auch umblättern.

Beispiel Overview:

Einzeleinsätze



Konvoi-Hinterhalt

Sie sind hinter den feindlichen Linien im Einsatz. Ein feindlicher Versorgungskonvoi nähert sich seinem Stützpunkt. Halten Sie ihn auf.

- Meine Missionen...
- Switzerland Mod...
- Trainingsmissionen...
- 01: Konvoi-Hinterhalt**
- 02: Reinigungsoperation
- 03: Lieferjunge
- 04: Kampfhubschrauber
- 05: Nächtllicher Angriff
- 06: Eiserne Faust
- 07: Alligator
- 08: Schlachtfelder
- 09: Kommandant
- 10: Die Bibliothek
- THE AH-64A

Stringtable.csv

In einer Stringtable.csv werden Strings (Zeichenketten) abgelegt, die man zur Ausgabe verwenden kann.

Strings sind ungefähr 32 bit groß (bei der Angabe bin ich mir nicht sicher) und es können alphanumerische Zeichen und Sonderzeichen in diesen Variablen abgelegt und ausgegeben werden. Bei einigen Programmiersprachen wie Delphi oder allgemein die Pascal-Sprachen ist das so, dass Eingaben des Benutzers Strings sind. Diese Eingaben werden dann entsprechend in die verschiedenen Datentypen umgewandelt und in die entsprechenden Variablen geschrieben.

Der Vorteil eines Strings ist, dass sowohl alphabetische als auch numerische Werte gespeichert und ausgegeben werden können. Das hat den Vorteil, dass man keine extra numerischen oder alphabetischen Variablen anlegen muss. Numerische Datentypen sind sowas wie Integer, short, byte, long, long long, float, double, long double. Alphabetischer Datentyp ist char(character).

Ein Integer und ein Char belegen zusammen mehr als 32 bit.

Kommen wir zurück zu den Strings. Ein String kann nicht als boolean Datentyp verwendet werden.

Schauen wir uns mal den Quellcode einer stringtable.csv an.

Auszug aus der Stringtable.csv von der Mission CODENAME 'SHILKA'
Language,English,German,comment

```
;------//Onloadmission\-----
```

STRD_onload,"Last preparations...","Letzte Vorbereitungen..."

STRD_avspres,"ACID VIPER STUDIOS present","ACID VIPER STUDIOS Präsentieren"

Das STR ist die Abkürzung für String. Es gibt unterschiedliche Attribute für die Strings. Attribute braucht ein String in der Stringtable nicht unbedingt.

Die Syntax eines String-Eintrags ist:

Language, English, German, French, Czech, comment

STRAttribut_variable, "Text in der ersten Sprache", "Text in der zweiten Sprache",...

Das language und alle Sprachen, die da hinter stehen, sagen aus, in welcher Installationssprache der Text ausgegeben werden soll. Das Programm sucht sich entsprechend die Texte heraus.

Kommentare werden hinter Semikola geschrieben. Informationen, die hinter einem Semikolon stehen, ignoriert der Rechner.

STRD_variable – Description.ext Ausgabe Attribut.

STRM_variable – Attribut für Mission.sqm Ausgabe

STRS_variable – Attribut für Ausgabe in sqs und sqf Skripten.

STRCAMP_variable – dabei bin ich mir nicht ganz sicher, aber das könnte die Abkürzung Campaign sein. Also speziell Strings für die Kampagnen Stringtable.csv.

STR_CFG_PAPABEAR," Callsign Papa Bear ändern" – dieser String erbt von der Cfg(Config) und in dem Beispiel kann der Name Papa Bear umgeschrieben werden, worauf man will. Dabei wird der eingegebene Text nicht in der Config Datei umgeschrieben. Dieser Text bleibt also lokal nur für die Mission.

Arbeiten mit Skripten

Eine Übersicht in Beispielen

Beginnen wir mit einer schnellen Einführung in das Kapitel Skripte.

Ich möchte euch die wichtigsten Sprachelemente anhand von einfachen Beispielen zeigen ohne euch mit Ausnahmen und Details zu überfluten. Ich möchte euch schnell an einen Punkt bringen an dem ihr eigene Skripte schreiben könnt und euren Zielen damit näher kommt. Ihr werden schnell feststellen, dass viele Aufgaben nur weiter Entwicklungen dieser hier aufgeführten Beispiele sind. Um diesem Ziel näher zu kommen müsst ihr jedoch zuerst verstehen welche Sprachelemente Arma und dessen Skriptsprache beinhaltet. Als Voraussetzung um ein Skript zu schreiben oder es zu verstehen ist es notwendig zuerst die Punkte Variablen, Berechenbarkeit von Variablen, Variable für Bedingungen, Arrays und Boolesche Befehle gelesen zu haben da sonst einige Strukturen der Skripte nur schwer zu verstehen sind.

Arrays

Ein Array ist ein Variablenfeld, bei dem Datenfelder gleichen Typs vorkommen. Stünde eine Variable nur allein, wäre sie nur ein Wert. Gewöhnlich bestehen sie aus alphanumerischen Zeichen (Strings=Zeichenkette), manchmal auch aus numerischen Zeichen (Ganzzahlen: short, Integer, long, long long, Fließkommazahlen: Float, Double usw.) oder auch aus alphabetischen Zeichen (Als Beispiel Char).

Beispiel:

Wert=[VariableimArray1,VariableimArray2]

Arrays werden immer in [eckige Klammern] gesetzt und durch ein Komma werden die Daten voneinander getrennt.

BOOLSche Befehle

BOOLSche Befehle sind sowas wie

this inflame **True**

Man kann sowas aber auch in eine Verzweigung einbauen wie in diesem Beispiel:

?Bedingung

Wenn die Bedingung bei einer Verzweigung **True** (Wahr; Der BOOLSche Wahrheitwert) ist, wird der **Then** Wert ausgeführt, Wenn die Bedingung **False** (Falsch bzw. unwahr; Ebenfalls ein BOOLScher Wahrheitwert) ist, wird der **Else** Wert ausgegeben. Eigentlich ja ganz einfach, aber bei Arma gibt es **keinen Else** wert. Ein sogenanntes BOOL (Boolean) ist immer ein Abfrage-Datentyp.

Berechenbarkeit von Variablen

Man kann auch mit Variablen, wie wir das aus der Mathematik kennen, rechnen. Was man alles verwenden kann, werde ich zeigen.

+ (integer,float,double)=Addition von den Datentypen
 - (integer,float,double)=Subtraktion von Datentypen
 * (integer,float,double)=Multiplikation von Datentypen
 / (integer,float,double)=Division von Datentypen
 % (integer,float,double)=Prozentberechnung von Datentypen
 ^ (integer,float,double)=Exponent von Datentypen
 Log (integer,float,double)=Logarithmus von den Datentypen
 LN (integer,float,double)=Logarithmus Naturalis von den Datentypen
 Random (integer,float,double)=Zufallswert zwischen 0 und Datentyp
 Sin (integer,float,double)=Sinus vom Winkel Datentyp
 Cos (integer,float,double)=Kosinus vom Winkel Datentyp
 Tan (integer,float,double)=Tangens vom Winkel Datentyp
 Sqrt (integer,float,double)=Wurzel aus Datentyp
 bei denen ich mir nicht so sicher war, hab ich weggelassen.

Variablen

Variablen sind nicht nur in der Mathematik von großer Bedeutung, sondern auch in der Informatik. Alle Programme heutzutage haben Variablen (Platzhalter) eingebaut. Variablen braucht man dafür, dass man nicht alles manuell eingeben muss. als Beispiel: c2 ist ein alphanumerisches Zeichen.

Man unterscheidet grundlegend zwischen den **Globalen** und den **_Lokalen Variablen**.

Globale Variable: `variable = 1`

Können sowohl im Editor als auch im Skript verwendet werden. Sie sind immer bekannt und bei der Namensvergabe ist darauf zu achten, ob dieser Name existiert und von wem darauf zugegriffen wird. Bei der Namenswahl sollte darauf geachtet werden, dass dieser mit einem kleinen Buchstaben beginnt. zB. `variable1` , `alarm`

Bei globalen variablen ist darauf zu achten wer auf diese zugreift und vor allem wer diese schreibt. Ansonsten kann dies zu erheblichen Problemen führen.

Lokale Variable: `_variable = 1`

Werden hauptsächlich in einer Funktion(Scripts) verwendet und können nur lokal ausgelesen werden. Die Namen können frei gewählt werden und in einem anderen Skript wieder verwendet werden. Bei der Namenswahl sollte darauf geachtet werden das dieser mit einem Unterstrich und dann mit einem kleinen Buchstaben beginnt. zB. `_variable1` , `_alarm`

Erste Schritte

Wo ist das Skript? Das Skript wird im Ordner der Mission positioniert. Es handelt sich dabei um eine reine Textdatei welche im Editor Notpad oder ähnlich erstellt wird und die Dateiendung *.sqz hat. Achtet bitte darauf das bei euch unter Favoriten / Ordneroptionen / Ansichten / Erweiterungen bei bekannten Dateitypen, ausblenden nicht aktiviert ist. Ansonsten werdet ihr Probleme mit den Skripten bekommen. Da die Endungen falsch sind. Starten eines Skript aus dem Editor.

Starten eines Skripts aus dem Editor

Um ein Skript zu starten, wird in einer Init Zeile, in einem Wegpunkt oder Auslöser der Aufruf eingetragen `[Parameter] exec "Name.sqs"` oder `[] exec "Name.sqs"`.

Beispiel:

`[] exec "hallo.sqs"` start ohne Parameter.
`[John,Anna] exec "hallo.sqs"` start mit John Anna als Parameter.

Starten eines Skripts aus einem Skript

Ein Skript in einem Skript aufrufen ist nichts anderes als ein Skript im Editor zu starten.

`[Parameter] exec "Name des Skript.sqs"`

Die Parameter können Werte aus der Funktion oder Globale Variablen sein. Auf diese Weise können Einzelne Schritte Kompakt ausgelagert werden.

Starten eines Skripts aus einer Funktion

Ein Skript wird in einer Funktion genau so gestartet wie aus dem Editor

`[Parameter] exec "Name des Skript.sqs"`

Die Parameter können Werte aus der Funktion oder Globale Variablen sein.

Starten einer Funktion aus einem Skript

Um eine Funktion in einem Skript zu starten ist es nicht nötig

`null = [Parameter] execVM "Name der Funktion.sqf"` zu schreiben.

Null = ist nur notwendig wenn diese Funktion aus dem Editor aufgerufen wird. In einem Skript ist das Automatisch. Dieser Funktion können nun Parameter übergeben werden. Dabei kann es sich um Globale Größen oder auch um lokale Parameter im Skript handeln.

`[Parameter] execVM "Name der Funktion.sqf"` Anders als bei einem Skript kann dann auch durch die Funktion ein Wert zurück an das Skript gegeben werden.

`_a = Funktion;`

anschließend befindet sich in der lokalen Variable `_a` der Funktionswert.

Die Arithmetik

Anweisungen und Strukturen

Ein Ausdrücke wie `_x = 1` oder `_a = getPos "player"` sind Anweisungen sobald ein Gleich Zeichen (=) zwischen der Variable und dem Wert steht. Anweisung werden nicht wie bei Funktionen mit einem Semikolon beendet. `_x = 1;` oder `hint "ArmA ist das Größte";` Sind jedoch auch OK. Das Semikolon ist also nicht zwingend Notwendig. Die runden Klammern (und) dienen dazu Anweisungen oder Vereinbarungen zusammen zu fassen. Die Aufgabe `_a = 5 * 5 +5;` Ergebnis `_a = 30` oder `_a = 5 * (5 +5);` Ergebnis `_a = 50` können so beeinflusst werden. Des weiteren ist es manchmal nötig Funktionen Werte zu übergeben. `_a = sin (35 +7);` Da sin einen Wert erwartet ist es von Nöten diesen zuerst zu berechnen. Daher die Klammern (und). Das Ende eines Skriptes wird durch `exit` gekennzeichnet.

Anweisungen

Anweisungen werden in einem Skript mit = realisiert. Variable = Wert.

`a = 5` oder `a = 5;`

`_a = 5` oder `_a = 5;`

Pause

Um ein Skript Pausieren zu lassen besteht die Möglichkeit die Funktion ~ zu starten. Diese erwartet einen Wert. (in Sekunden Typ float) Es können also alle Gleitpunktwerte an diese Funktion Übergeben werden. Das Skript Pausiert an der Stelle dann s lange bis der Übergebene Wert erreicht wurde.

~2.65

Das Skript wartet 2.65 Sekunden.

Pause bis Bedingung erfüllt ist

Um das Skript pausieren zu lassen bis ein Ereignis eingetroffen ist bestehen die Möglichkeiten dies über @ Bedingung zu machen. Ein Skript soll pausieren bis die Einheit Sworn tot ist.

@ (not alive Sworn) wartet jetzt im Skript bis Sworn tot ist.

Um ein Skript Pausieren zu lassen bis seine Variable wieder auf true gesetzt ist kann dies mit @Variable erfolgen. Variable steht hier als Platzhalter und kann durch jede Globale Variable ersetzt werden. Es muss sich um eine globale Variable handeln da das Skript Pausiert und keine lokalen Variablen gesetzt werden können.

Vergleiche

Unter Vergleichen versteht man Operationen mit zwei oder mehreren Werten.

wahrheitswert1 **and** wahrheitswert2 (Logische und Anbindung)

wahrheitswert1 **or** wahrheitswert2 (Oder Verknüpfung)

wahrheitswert1 **= =** wahrheitswert2 (wahrheitswert1 ist gleich wahrheitswert2)

wahrheitswert1 **<** wahrheitswert2 (Kleiner als wahrheitswert1)

wahrheitswert1 **<=** wahrheitswert2 (Kleiner als wahrheitswert1 bzw. gleich)

wahrheitswert1 **in** wahrheitswert2 (wahrheitswert1 vorhanden in wahrheitswert2)

wahrheitswert1 **!=** wahrheitswert2 (Ist Ungleich)

wahrheitswert1 **>** wahrheitswert2 (wahrheitswert1 größer als wahrheitswert2)

wahrheitswert1 **>=** wahrheitswert1 (wahrheitswert1 größer als wahrheitswert2 bzw. gleich)

In der Regel werden die Vergleiche in runde Klammern (und) geschrieben damit diese zusammen gefasst sind.

Vorrang und Assoziativität der Operatoren

Operatoren	Assoziativität
() []	von links her
! ~ ++ --	von rechts her
* / %	von links her
+ -	von links her
<< >>	von links her
< <= > >=	von links her
== !=	von links her
&	von links her
^	von links her
	von links her
&&	von links her
?:	von rechts her
= += -= *= /= %= ^= = <<= >>=	von rechts her
,	von links her

Sprung Marker

In einem Skript kann in den Zeilen hin und her gesprungen werden. Hierbei unterscheidet man die Marke zu der gesprungen wird, und der Sprungpunkt, von der aus gesprungen wird. Sprungpunkte werden mit goto "Marke" ausgezeichnet. Kommt das Skript an diese Stelle springt das Skript zu der ausgezeichneten Marke. Die Marke selber wird mit Raute Name ausgewiesen. #Marker. Es können beliebig Sprung Marken in einem Skript verwendet werden.

```
#Marker2
goto "Marker1"
Befehlx
#Marker1
goto "Marker2"
```

In diesem Beispiel springt das Skript zuerst zur Marker1 Position um dann sofort zur Markerposition2 zu springen. Somit wird der Befehlx nie ausgeführt.

Schleifen

Schleife bis Bedingung erfüllt ist

Eine Schleife kann zB. so lange ausgeführt werden bis eine Bedingung zu trifft.

```
#Marker
~1
?alive Sworn: goto "Marker"
hint "Sworn lebt nicht mehr"
exit
```

Diese Schleife wird nun einmal in der Sekunde abfragen ob die Einheit Sworn lebt, wenn ja wird sie zur Marker Position zurückkehren. Dies wiederholt sich so lange bis die Einheit tot ist.

Ausgaben auf den Bildschirm

Um Skripte zu testen und dessen Werte zu auszugeben, verwendet den Befehl hint "Das Skript ist an der Stelle" oder `hint format ["Wert = %1",_i]`; In diesem Beispiel würde das Skript den Wert von der Variablen `_i` ausgeben. Sehr nützlich wenn mit Werten in einem Skript gearbeitet wird.

Auslesen von Parametern

Die an ein Skript übergebenen Parameter werden in Variable, lokale Variable im Skript übergeben. Dazu werden diese der Reihe nach ausgelesen. Dies passiert meist zu Beginn eines Skriptes. [John, Anna] exec "hallo.sqs"

Quellcode:

```
_a = _this select 0;
_b = _this select 1;
```

In `_a` steht jetzt John, und in `_b` steht jetzt Anna. mit diesen Variablen kann jetzt gearbeitet werden als ob da die Namen John und Anna stehen würden.

Erklärungen hinterlassen

Um Erklärungen in Skripten in Textform zu hinterlegen wird vor dem Text ein Semikolon gesetzt. Somit kann jeder Befehl dokumentiert werden. ; hier kann der Text stehen

Abfragen von Werten

Es gibt die Möglichkeit Werte wie Schaden Treibstoffmenge etc abzufragen. Hierzu gibt es in ArmA feste Funktionen. In diesen Beispielen wird in die Lokale Variable (_a) der jeweilige Wert geschrieben.

_a = Speed Sworn	Funktionswert Geschwindigkeit in Km/h
_a = Skill Sworn	Funktionswert Fähigkeit (1 bis 0)
_a = GetPos Sworn	Funktionswert Array Position (XZY)
_a = GetDammage Sworn	Funktionswert Schaden aus (1 bis 0)
_a = GetDir Sworn	Funktionswert Blickwinkel (0 bis 360)
_a = Time	Funktionswert Zeit in Sekunden vom Missionsbeginn
_a = DayTime	Funktionswert Tageszeit (0.000 bis 24.000)
_a = AccTime	Funktionswert aktuelle Spielgeschwindigkeit
_a = Benchmark	Funktionswert Systemspezifischen Benchmarkwert
_a = Alex knowsabout Sworn	Funktionswert wissen von Alex über Sworn
_a = overcast	Funktionswert aktueller Wetterwert
_a = overcastForecast	Funktionswert zukünftiger Wetterwert
_a = Fog	Funktionswert aktueller Nebelwert
_a = FogForecast	Funktionswert zukünftiger Nebelwert

Setzen von Werten

Das geht natürlich auch anders herum. Einer Einheit können Werte gesetzt werden.

Name der Einheit SetDamage 1 Tötet die Einheit

Name der Einheit Setfuel 1 Setzt die Treibstoffmenge

Dies ist mit allen Werten Möglich (siehe Blickrichtung Waffen). Die Befehle erhalten ein Set + Befehl Wert und werden somit als Zuweisung bearbeitet.

Ausgabe von Werten

Die so gewonnen Werte können in Funktionen etc. jetzt genutzt werden. Oder auf den Bildschirm ausgegeben werden. Dies kann als **hint TitleCut** oder Funkspruch erfolgen. Dazu wird im Text die Variablenmarkierung gesetzt **%1** für die erste Variable **%2** für die zweite Variable. Die Abfolge ist hierbei egal. Die Nummer bezieht sich auf die Position an der die Variable vereinbart wird.

Name des fliegenden Helikopters Heli1

Beispiel: **hint format ["Speed = %1\nWinkel = %2",speed Heli1,GetDir Heli1];**
exit

Jetzt erscheint eine Nachricht mit Speed = Wert und in der Zeile darunter Winkel = Wert.
\n erzeugt einen Zeilen Umbruch.

Scriptbeispiele:

Ich hoffe das diese kleinen Beispiele dem einen oder anderen beim Einstig in den Bereich Skripte helfen können.

Unterstützung Rufen

Wer kennt das nicht, man befindet sich in der schönsten Schlacht und versucht sich die Straße weiter zu kämpfen, und dann steht da ein riesiger T72 im Weg. Keine M136 in Reichweite und auch keine Möglichkeit sich um das Monster herum zu schleichen. Was würde man jetzt dafür tun dem nächsten Piloten die Zieldaten geben zu können und in der Deckung abzuwarten bis das Monster in brennende Einzelteile zerlegt ist.

Hier ist eine Idee um das umzusetzen. Zuerst gebt dem Soldaten ein Aktionmenü welches dann ein Skript startet. `Aktion1 = Alex addAction ["Hilfe rufen", "hilfe.sqs"];` oder das Skript wird über einen Funkauslöser aufrufen. Je nach Wunsch des Missionserstellers.

Dieses macht dann folgendes:

Es ließt den Namen der nächsten Lufteinheit, gibt dieser Einheit die Order sich zu der Position des Rufers zu begeben. Voraussetzung hierfür ist jedoch dass eine Lufteinheit existiert. Das Skript wird dann im Missionsordner erstellt. (hilfe.sqs) und enthält den Quellcode:

```
_a = position player nearObjects ["Air",2000];
_a DoMove GetPos player;
exit
```

Kleines Skript große Wirkung. Aber gute Dinge müssen ja nicht immer sehr kompliziert sein.

Speed

Ich habe vor Jahren einen Film mit dem Namen Speed gesehen. Es gab da viele aber ich meine den ersten Teil. In dem Film fährt ein Bus mit einer Bombe durch LA. Wenn er langsamer als 45mph fährt explodiert dieser. Ein einfaches Skript und ihr werdet den Film mit neuen Augen sehen. Erstellt eine Einheit „Bus“ mit dem Namen Bus und lasst den Spieler die Kontrolle übernehmen. In dessen Initzeile ruft ihr mit `[] exec "speed.sqs"` das Skript auf. Im Missionsordner erstellt ihr nun die Datei speed.sqs und fügt folgendes ein.

Quellcode:

```
@(speed Bus > 45)
hint "Bombe ist scharf"
@(speed Bus < 40)
Bus setdamage 1
exit
```



Das FallschirmSkript

Parameter: 2.

Aufruf im Spiel : [Name der Gruppe die springen soll, Name des Helikopters] exec "fallschirm.sqs" Zum Beispiel : [Grp1,Luft1] exec "fallschirm.sqs"



fallschirm.sqs

```

; *****
;
; ** ArmA Fallschirm Sprung Script File **
; *****

```

```
_gruppe = _this select 0
```

```
; Übergabe Position 0 = _gruppe
```

```
_luft = _this select 1
```

```
; Übergabe Position 1 = _luft
```

```
_aunits = units _gruppe
```

```
; Übergibt den Gruppen Namen
```

```
_i = 0
```

```
; Variable _i auf 0
```

```
_j = count _aunits
```

```
; Zählt die Gruppe = _j
```

```
#Schleife1
```

```
; Schleifenmarker
```

```
(_aunits select _i) action ["EJECT", _luft]
```

```
; Gibt die Order, auszusteigen
```

```
unassignvehicle (_aunits select _i)
```

```
; Streicht die unit aus dem Cargo
```

```
_i = _i + 1
```

```
; addiert 1 auf _i
```

```
~1
```

```
; Pause für eine Sekunde
```

```
? _j > _i : goto "Schleife1"
```

```
; Abfrage ob _j größer _i, wenn
```

```
; ja springt das Skript zum
```

```
; #Marker zurück.
```

```
exit
```

```
; Beendet das Skript
```

In diesem Skript wurden gleich eine ganze Reihe an Funktionen verwendet. Siehe hierzu Parameter und Skript Befehle. Die Fallschirmspringergruppe kann am besten über `{_x moveincargo Luft1} foreach units Grp1`; in das Flugzeug gesetzt werden. Grp1 ist der Name des Anführers der Sprung, Luft1 der Name der Lufteinheit.



Das GPS Marker Skript1

Dies ist die einfache Form eines GPS Marker Skript. Es wird in einer Init Zeile gestartet und bedarf keinerlei Parameter. Der Nachteil ist jedoch, dass der Marker im voraus im Editor gesetzt werden muss und dieses Skript nur mit einer Einheit läuft. Hierbei wird für jeden Marker ein eigenes Skript benötigt.



Gps1.sqs

```

; *****
;
; ** ArmA GPS1 Script File **
; *****

#Start                               ; Schleifenmarker
"Marker1" setMarkerPos getpos Alex   ; Setzt den Marker Name marker1
                                       ; auf die Person von der Einheit
                                       ;Name Alex
~2                                     ; Pause für zwei Sekunden
goto "Start"                          ; das Skript springt zum #Marker

```

Das GPS Marker Skript2

Verbessertes Marker Skript mit Parametern (Name des Markers, Name der Einheit)
Dieses Skript arbeitet wie GPS1. Es werden lediglich die Namen übergeben, was dieses Skript für mehrere Marker nutzbar macht. Um dieses zu starten, wird in der Init Zeile einer Unit

[this,Marker2] exec "GPS2.sqs" eingefügt oder wenn es nicht aus der gemeinten Unit gestartet wird [Name der Einheit, Name des Markers] exec "GPS2.sqs"



Gps2.sqs

```

; *****
;
; ** ArmA GPS2 Script File **
; *****

_eneit = _this select 0
_marker = _this select 1

#Schleife
? (not alive _eneit): goto "Ende"

_marker setMarkerPos getpos _eneit
-1
goto "Schleife"

#Ende
deleteMarker _marker
exit

```

Das GPS Marker Skript3

Verbessertes Marker Skript. In dieser Fassung wird ein bestehender Marker in Form Farbe Ausrichtung sowie Größe angepasst und folgt dem Spieler. Der Name der Einheit = Alex. Name des Markers = Marker1 . Experten können auch mit Parametern arbeiten, Werte übergeben. Es besteht auch die Möglichkeit den Marker mit createMarker [name, position] oder createMarkerLocal [Name, Position] selber zu erschaffen damit dieser nicht erst erstellt werden muß.

```
; *****
;
; ** ArmA Script File **
; *****

"Marker1" setMarkerSize [1, 1]           ;größe Einstellen
"Marker1" setMarkerShape "ICON"         ;Bestimmung des Typ in Verbindung mit
                                         ;camcreatemarker sinnvoll

"Marker1" setMarkerType "Arrow"         ;Typ des Markers wählen (Pfeil)
"Marker1" setMarkerColor "ColorBlack"   ;Farbe des Markers bestimmen
"Marker1" setMarkerDir 0                 ;Ausrichtung des Markers (0°)
"Marker1" setMarkerText "GPS you."      ;Text welcher be idem Marker angezeigt wird

#setzmarker                               ;Schleifenstart
_d = direction Alex                       ;Auslesen der Ausrichtung von Alex
"Marker1" setmarkerpos getpos Alex        ;Marker auf Alex setzen
"Marker1" setMarkerDir _d                 ;Marker ausrichten
? not alive Alex: goto "loeschemarker"   ;Abfrage ob Alex lebt
goto "setzmarker"                         ;Schleifenende

#loeschemarker
deleteMarker "Marker1"                   ;lösche den Marker
exit
```

Evakuierungs Punkt aussuchen

Ein kleines Szenario.

Eine dunkle Nacht das Mondlicht schimmert grünlich im Nachtsichtgerät. Der Anführer meldet sich über Funk beim HQ mit der Nachricht. Operation erfolgreich. Landezone Sirra ist unsicher. Erbittle Abholung bei AE5 347.over. Danach markiert der Spieler einen Punkt auf der Karte und der Helikopter fliegt diesen an und geht an diesem Punkt 1m über der Erde in den Schwebeflug. Die Spieler klettern in den UH 60 und dieser fliegt ab. Beim Einsteigen vernimmt der Anführer noch einen dumpfen Knall aus der Ferne und lächelnd meint er: Treibstofftanks brennen in der Nacht besonders schön.

Und die Umsetzung in ArmA.

- Erstellen der Einheit Spieler. Name **Delta1**
- Erstellen des Helikopters ohne Wegpunkte. Name **Luft1**
- Erstellen eines Funkauslösers. Alpha etc. startet das Skript. **"einsammeln.sqs"**
- Erstellen eines Markers. Diese ist für den Sammelpunkt. Name **Sirra**

Das Skript setzt jetzt dann Marker an den Punkt an den Punkt der vom Spieler auf der Karte durch Mausklick gewählt wird. Der Helikopter erhält die Order diesen dann an zu fliegen, dort in den Schwebeflug zu gehen und wenn der Spieler eingestiegen ist, oder getötet wurde, zurück zu fliegen. Übergabe der Parameter Name des Soldaten, Name des Markers, Name der Lufteinheit.

Quellcode einsammeln.sqs:

```

_unit = _this select 0;
_marke = _this select 1;
_air = _this select 2;

LZ = "HeliHEmpty" createVehicle [0,0,0];

target = true;
[west,"HQ"] SideChat "Bitte auf der Map die Zielkoordinaten bestimmen"
onMapSingleClick "LZ setpos [_pos select 0, _pos select 1 , _pos select 2];target=false"
@ !target
~1
_marke setmarkerpos getPos LZ
[west,"HQ"] SideChat "Sind auf dem Weg."
_air DoMove getMarkerPos _marke

@70 > (_air Distance _unit)
[west,"HQ"] SideChat "Setzen zur landung an."
_air flyinheight 1
@(_unit in _air) or (not alive _unit)
_air flyinheight 50
_air DoMove [0,0]

```

Dieses Skript sieht etwas komplizierter aus. Es werden 3 Parameter übergeben, Ein Heliport LZ erstellt, dieser wird an die Position des Mapklick gesetzt, der Helikopter startet und fliegt zur LZ, wenn der Spieler Delta1 dichter als 70m am Heli ist sinkt dieser, und wenn der Spieler tot oder im Heli ist fliegt der Heli zu den Koordinaten 0,0.

Artillerie

Die Artillerie bestimmt ein Schlachtfeld wie keine andere Waffe. Hier werden Beispiele gezeigt wie man Artillerie in ArmA nutzen kann. Jedoch sollte der Leser wissen was Artillerie ist und wie diese eingesetzt wird. Artillerie wird auf zwei verschiedene Arten eingesetzt. Diese unterscheiden sich in der Flugbahn. Im Prinzip gibt es für jede Entfernung zwei Abschusswinkel. Diese werden als direkt und indirekt bezeichnet. Die direkte Form der Beschuss, ist die flache und schnellere Flugbahn. Dabei wird das Ziel direkt anvisiert, die Abweichung berechnet als Funktion zur Distanz. In der Praxis steht der User jedoch schnell vor dem Problem das sein Ziel von Bergen oder Hügeln verdeckt steht und ein direkter Beschuss nicht möglich ist. Hier hilft nur der indirekte Beschuss: Dabei wird das Geschoss auf einer hohen Flugbahn auf das Ziel abgeschossen. Der Aufschlagwinkel ist steiler als bei dem direkten Beschuss. Je nach Position des Ziels liegt es am Spieler direkten oder indirekten Beschuss auszuwählen.

Hier werden verschiedene Skripte gezeigt mit denen Artillerie realisiert werden kann.

Artillerie1

In diesem Beispiel wird eine Position über einen Mapclick festgestellt, über dieser Position werden dann Granaten erstellt welche dann da vom Himmel fallen. Das ist die einfachste Lösung.

Dazu werden zwei Skripte gebraucht. Das erste ist ein „einfaches klicke auf die Map und übergebe die Position an Skript Nummer2“. Um das Skript 1 zu starten einfach einen Funkauslöser oder vergleichbares einsetzen.

```
; *****
; ** ArmA Script File **
; *****
taget = true
ForceMap true
[west,"HQ"] SideChat "Bitte auf der Map die Zielkoordinaten bestimmen" ;
ziel = "HeliHEmpty" CreateVehicle [0,0]
onMapSingleClick "ziel setpos _pos;taget = false"
@!taget
ForceMap false
~1
player SideChat "Sende die Zielkoordinaten."
[ziel] exec "ari1.sqs"
Exit
```

In diesem Skript wird der Benutzer aufgefordert auf der Karte eine Position zu wählen. Diese wird dann an Skript2 weitergegeben.

Das Feuerskript. Parameter `[Name des Ziel] exec "ari1.sqs"`

```
; *****
; ** ArmA Script File **
; *****
_ziel = _this select 0
_x = Getpos _ziel select 0
_y = Getpos _ziel select 1
_j = 0;
#lol
_j=_j+1
Granate1="Sh_105_HE" CreateVehicle [_x+10-random 20,_y+10-random 20,200+50-
random 100]
?_j < 5 : goto "lol"
exit
```

In diesem Skript wird auf die x und y Länge und Breite ein Wert addiert. Das bewirkt eine streuen der Granaten. Dasselbe wird mit der Höhe der Granaten gemacht. Das sorgt für einen versetzten Einschlag. So wirkt es als würde die Ari wirklich schießen.

Artillerie mit Geschützen

Diese Artillerie funktioniert in etwa wie das erste Beispiel. Es werden lediglich Geschütze auf die Karte gestellt, diese feuern dann in die Luft, ArmA löscht das Projektil nach 20 Sekunden, und es werden neue im Zielgebiet erstellt. Weitergehend wird überwacht in welchem Zustand sich die Artillerie befindet. Experten können das auch mit der Abfrage ob die Geschütze existieren und Munition haben verbinden.

Kamera Camera

Viele Missionen gerade im Singleplayer lassen was das Ambiente angeht einiges zu wünschen übrig. Nun ja. Der Editor verfügt nicht nur über die Möglichkeit eine Mission zu bauen sondern auch über einen Vorspann und einige Nachrufe. Doch was ist ein Vorspann Nachspann? Es handelt sich hierbei nicht um ein spielbares Stück einer Mission. Es ist mehr ein Film. Doch jeder Film benötigt ja nun eine Kamera. In ArmA gibt es nicht die Möglichkeit AVI Dateien einzubauen, und das ist auch gut so. Die ganzen Kamera Bewegungen werden im Editor vorbereitet und dann meist durch ein Skript gesteuert. Das Ganze ist in diesem Manual anhand eines Beispiel erklärt.

1. Erstellen einer Kamera, aus einem schwarzen Bild heraus.

- a) Bild schwarz färben. `TitleCut ["Meine Mission", "Black In", 5];`
- b) Kamera an Position XYZ erstellen.

```
_Cam1 = "camera" camcreate [X,Y,Z];
```

- c) Kamera einen schwarzen Hintergrund geben.

```
_Cam1 cameraeffect ["internal", "back"];
```

```
// a) Erstellt einen Schriftzug mit Mein Text.
```

```
// b) Erstellt eine Kamera an den World Koordinaten XYZ. Der Name dieser ist _Cam1.
```

```
// c) färbt den Bildschirm schwarz.
```

2. Der Kamera ein Ziel zu weisen.

- a) `_Cam1 CamSetTarget target1;`

```
// a) Die Kamera (Name _Cam1) richtet sich auf das Ziel mit dem Namen target1. Target1 kann eine Einheit oder eine Logik ein Marker etc. sein.
```

// jetzt haben wir eine Kamera die steif auf das Ziel gerichtet ist und diesem folgt. Ich persönlich bevorzuge Kamerasequenzen in denen die Kamera über das Land schweift und sich dann auf ein Ziel ausrichtet. Das wollen wir jetzt mal machen.

3. Der Kamera ein weiteres Ziel zu weisen.

- b) `_Cam1 CamSetTarget target2;`

4. Zeit festlegen damit die Kamera nicht gleich auf target2 zu springt.

- a) `_Cam1 CamCommit 15;`

```
// jetzt schwenkt die Kamera langsam von target1 zu target2. 15 ist die zeit in Sekunden.
```

Damit das Skript nicht weiterläuft kann diesem eine Pause mit dem Selben Zeitwert gegeben werden oder der Befehl `@CamCommitted _Cam1` genutzt werden.

5. Kamera eine Position an einem Ziel vermitteln. Die Eingabe der Werte erfolgt in XYZ Werten.

- a) `_Cam1 CamSetRelPos [50,20,10];`

```
//Die Kamera bewegt sich jetzt zu einer Position welche X = 50m weiter rechts, Y = 20m weiter Nördlich, Z = 10m über dem Boden. Dieser Bewegung kann jetzt wieder eine Zeit mitgegeben werden. _Cam1 CamCommit 7; Die Kamera ist also nach 7 Sekunden am Ziel.
```

Befehl `@CamCommitted _Cam1` hinzufügen damit die Kamera nicht gleich weiter im Skript macht.

Jetzt wissen wir wie wir eine Kamera erstellen, sie positionieren, sie ausrichten und auf Position „fahren“. Es entstehen bereits schöne Sequenzen wenn die Position der Kamera mehrfach sich ändert und dabei dasselbe oder ein anderes Ziel erfasst wird. Um das Skript zwischen einzelnen Sequenzen zu Stoppen, `~5` einfügen. Das Kameraskript stoppt an dieser Stelle wie jedes andere Skript auch.

Um eine fertige Kameraführung jetzt zu beenden empfiehlt es sich den Bildschirm erneut schwarz aus zu blenden. Den Befehl kennen wir und haben ihn zu Beginn benutzt.

```
TitleCut ["Ende", "Black In", 2];
```

6. Kamera löschen

a) `camdestroy _Cam1`

//Löscht jetzt die Kamera.

Weiter Spielereien und Befehle für die Kamera.

Zoom +/-

```
_Cam1 CamSetFOV _zoom;
```

für `_zoom` wird ein Wert vom Typ float eingefügt. 0.1 oder 2.6 je nach Zoom.

Digitaler Zoom

Einen „dynamischen“ Zoom kann der User mit einer Schleife erstellen. (siehe Skript Funktion Schleifen ArMA Manual).

```
_zoom = 1
_digit = 0.001
_zoom2 = 0.1
#schleife
_zoom = _zoom - _digit
_cam1 CamSetFOV _zoom
_cam1 CamCommit 0
~0.02
? _zoom > _zoom2 : goto "schleife"
```

// Der Start und End Zoom wird gesetzt. In der Schleife wird der Zoom (`_digit`) abgezogen und der Zoom neu gesetzt. Die Schleife endet sobald der Wert kleiner ist als der `_zoom2`.

Man darf nicht vergessen, dass es sich hierbei, auch wenn es sehr fein ist um einen digitalen Zoom handelt. Das lässt sich jedoch auch gezielt einsetzen indem man einzelne Zoom Stufen nutzt. Hierzu wird der `_digit` im Wert stark vergrößert. Das hat dann den Effekt als ob man bei einem Fernglas den Zoom von 12x auf 24x erhöht. Es können so interessante Sequenzen entstehen.

Musik starten

Die schönsten Kameraführungen werden interessanter wenn Musik das Bild untermahlt.

Befehl: **Playmusic** "Name des Stück"

Es können die von ArMA bereit gestellten Musikspuren genommen werden oder eigene aufgerufen werden. siehe ArMA Manual Musik einbinden.

Musik beenden

Befehl: Zeit für die Änderung **fadeMusic** Lautstärke;

2 **fadeMusic** 0;

Setzt also in 2 Sekunden die Lautstärke auf 0 runter.

Kamera im Gelände fixieren

mit **CamSetTarget** kann der Kamera ein Ziel zugewiesen werden. Es brauch jedoch kein direktes Ziel zu sein. Es kann auch als Position eine Koordinate übergeben werden. [X,Y,Z]

_cam1 CamSetTarget [100,200,50]

Die Kamera würde sich jetzt auf 100m vom Westlichen Rand 200m vom südlichen Rand und auf eine Höhe von 50m ausrichten.

Variablen setzen

Um Ereignisse im Skript zu starten können wie in jedem Skript Variablen gesetzt werden.

Diese werden einfach in das Skript eingefügt. zB. go = true oder um ein Intro, einen Nachspann zu beenden eine Variable setzen, welche dann einen Auslöser welcher auf ende gesetzt ist auslöst.

Kamera über einem Objekt Erstellen.

_Cam1 = "camera" camcreate [(GetPos target1 Select 0),(GetPos target1 Select 1), 50];

In diesem Fall wird die Kamera über target1 erstellt. In der Höhe von 50m. Der Eintrag in den [Klammern] bestimmt die XYZ Position. Siehe Array ArMA Manual.

Kamera an eine Position beamen

Um eine Kamera zu teleportieren gibt es viele Möglichkeiten. Der Befehl hierzu ist Name der Kamera **camsetpos [X,Y,Z];**

oder wenn sie an die Position eines Einheit soll

Name der Kamera camsetpos getpos Name der Einheit;

Jedoch sieht das sehr abgehackt aus. Daher empfiehlt es sich den Bildschirm zu färben.

TitleCut ["Meine Mission","BLACK Out"]

~3

Name der Kamera camsetpos getpos Name der Einheit;

TitleCut ["Meine Mission","BLACK IN"]

In diesem Beispiel wird der Bildschirm schwarz die Kamera wird versetzt und das Bild wird wieder hell.

Kamera an ein Objekt heften.

Sequenzen mit fahrenden oder fliegenden Einheiten verlangen es manchmal, das der Kameramann mitläuft oder das ihm Flügel wachsen. (Ein echter Superman also) Dies kann mit einer Schleife und dem beam Befehl erfolgen.

```
_frame = 0.02
_zeit = 10
#schleife1
_zeit = _zeit + _frame;
_Cam1 camsetPos [(GetPos target1 Select 0) + 10,(GetPos target1 Select 1) + (-20), (GetPos
target1 Select 2) + 5];
~_frame
?_zeit < _ende: goto "schleife1"
```

In diesem Fall wird die Kamera immer an eine Position 10m weiter im Osten. 20m weiter im Süden und 5m höher gesetzt.

Ich halte diese Methode jedoch für etwas unsauber.

Methode zwei

Kamera einmal in Position setzen und dann die Geschwindigkeit anpassen.

```
_Cam1 camsetPos [(GetPos target1 Select 0) + 10,(GetPos target1 Select 1) + (-20), (GetPos
target1 Select 2) + 5];
_frame = 0.02
_zeit = 10
#schleife1
_a = velocity target1
_b = _a select 0
_c = _a select 1
_d = _a select 2
Sworn2 setvelocity [_b,_c,_d];
~_frame
?_zeit < _ende: goto "schleife1"
```

Bei vielen Schleifen in einem Skript bitte darauf achten das diese unterschiedliche Namen haben.

Kamera Effekte

für

"PLAIN", "PLAIN DOWN", "BLACK", "BLACK FADED", "BLACK OUT", "BLACK IN", "WHITE OUT" or "WHITE IN".

Beispiel einer Kamera Führung.

Quellcode

Beispiel einer Kamera führung.

```
TitleCut ["Meine Mission","BLACK IN"]
_cam1 = "camera" camcreate [1000,1000,100]
_cam1 camCommitPrepared 0
_cam1 cameraeffect ["internal", "back"]
_cam1 camPrepareFOV 1
_cam1 CamSetTarget target1
_cam1 CamSetRelPos [5,50,10]
_cam1 CamCommit 10
@CamCommitted _cam1
_cam1 CamSetRelPos [20,10,2]
_cam1 CamCommit 5
@CamCommitted _cam1
~2
_cam1 CamSetTarget target2
_cam1 CamCommit 10
@CamCommitted _cam1

_zoom = 1
_digit = 0.01
_zoom2 = 0.1
#schleife
_zoom = _zoom - _digit
_cam1 CamSetFOV _zoom
_cam1 CamCommit 0
~0.02
? _zoom > _zoom2 : goto "schleife"
~3
_cam1 CameraEffect ["Terminate","Back"]
camdestroy _Cam1
exit
```

Es sollte vielleicht erwähnt werden das Intros und Nachspann nur im SP laufen. In MP Missionen müssen die Kamerasequenzen in der Hauptmission mit eingebunden sein.

Die Lippen

Was einem perfekten Video nicht fehlen darf ist das Sprechen von Soldaten. Dazu ist es jedoch nötig den Abstand der Lippen zu bestimmen. Die große Kunst dabei ist das synchronisieren von Sound und Lippen. Jeder Sprachfile kann mit einer Lippen.txt Datei versehen werden. Die Werte in dieser Datei stehen beschreiben den Öffnungswert des Mundes. 0 steht für geschlossen 3 für weit geöffnet. Was muss in die Lippendatei?

- Die Taktfrequenz wird in Sekunden angegeben. 0.01 = 1 Hundertstel Sekunde. Alle 0.01 Sekunden list das System jetzt den nächste Lippenabstand aus.
- Zeit der Mundeinstellung 1.54 steht demnach für 1 Sekunde und 5 Zehntel 4 Hundertstel

Quellcode einer Lippdatei bei 0.03 Mund öffnen und fast wieder schließen.

```
frame= 0.03
0.00          0
0.03          0.25
0.06          0.5
0.09          0.75
```

0.12	1
0.15	1.25
0.18	1.5
0.21	1.75
0.24	2
0.27	2.25
0.30	2.5
0.33	2.75
0.36	3
0.39	2.75
0.42	2.5
0.45	2.25
0.48	2
0.51	1.75
0.54	1.5
0.57	1.25
0.60	1
0.63	0.75

Camera.sqs

Das bekannteste Skript ist das Camera Skript. Dieses bedarf keiner externen Datei. Es ist fester Bestandteil von ArmA. Es kann jederzeit und Überall gestartet werden. Hierzu wird einfach der aufruf **this exec "camera.sqs"** in eine Initzeile einer Einheit geschrieben. Jetzt wird bei Beginn der Mission eine Objekt freie Camera erstellt, welche frei über der Insel geflogen werden kann.

Steuerung der Camera:

W forward	8 tilt upward
E fast forward	2 tilt downward
A left	D right4 rotate left
S backwards	Q up6 rotate right
Z down	
+ zoom in	- zoom out
L get rid of the crosshair	
/ (on NumPad) target nearest object OR position on ground	
Space Bar also targets nearest object OR position on ground	
Del Turn on/off floating mode	

Cameraflug aufzeichnen

clipboard.txt

Es besteht die Möglichkeit einen Cameraflug aufzuzeichnen. Dazu einfach den Feuerknopf drücken. In dem Ordner

**C:\Dokumente und Einstellungen\Alex.Sworn\Lokale
Einstellungen\Anwendungsdaten\ArmA**

wird die clipboard.txt angelegt. In dieser werden jetzt alle Steuerungen der Camera aufgezeichnet und können danach über ein Skript etc aufgerufen werden. Wird hiervon Gebrauch gemacht wird bei der aktivierung ein Log eingetragen. Dadurch ist es möglich einige Sequenzen herraus zu nehmen. Logbeispiel:

```

;=== 22:11:02
_camera camPrepareTarget [684.73,98559.95,-27662.98]
_camera camPreparePos [2545.07,2486.85,9.91]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera

```

MP Skripte

Skripte werden im Allgemeinen von allen Spielern ausgeführt. Das führt zu Problemen wenn zB Einheiten erstellt. Hierbei kann es dazu kommen das nicht eine Einheit sondern Einheiten in der Anzahl des Spielers erstellt werden. Problemlösung: Um das zu verhindern muß sicher gestellt werden das ein Skript nur auf dem Server ausgeführt wird. Dazu erstellt eine Logik des Spiels mit dem Namen `server_sworn`. (Der Name ist natürlich frei wählbar). Und zu Beginn eures Skript fügt ihr eine Abfrage ein.

```
?local server_sworn: goto "ausführen"
exit
#ausführen
```

Jetzt wird bei dem Start eines Skript überprüft ob es um die Server Maschine handelt oder um die Spieler Maschine. Der Server führt das Skript aus und springt nach der Abfrage zu `#ausführen`, die Spieler Maschine geht zu `exit`.

Eine Alternative hierzu ist auch noch die Möglichkeit zu Beginn des Skriptes eine **globale Variabel** zu setzen. Und diese abzufragen. (in der `init.sqs` muss hierbei zu Missionsstart die Variable auf `true` gesetzt werden.)

```
?testvariable == true: goto "ausführen"
exit
#ausführen
testvariable = false
```

Die glorreichen Vier

Bei MP Missionen besteht ein kleines aber gravierendes Problem. Jeder Rechner führt jedes Skript aus. Dabei entstehen einige gravierende Probleme mit der Erstellung von Objekten oder vom Umsetzen globaler Variablen oder Aktionen. (Erstellen einer Explosion, setzen einer Variablen etc.) Daher ist es wichtig die einzelnen Bereiche zu unterscheiden. Es handelt sich hierbei um, **lokale** und **globale** Server Skripte und Ausführungen, oder um **lokale** und **globale** Client Skripte und Ausführungen.

Lokal Server = Berechnen von Werten, Abfrage von Auslösern etc.

Global Server = Ausgabe der berechneten Werte setzen von globalen Größen (Verwalten der Einheiten).

Lokale Client = Ausführen von für den Client vorgesehenen Skripten.

Global Client = Übertragen von Aktionen die der Spieler aufruft. (Drückender Feuertaste. Bewegungen etc.) Erstellen von Einheiten in Skripten wenn diese **lokal** ausgeführt werden. (Fehlerquelle)

Im Grundsatz sollte jedes Skript welches **globale** Bedeutung hat (Fallschirmspringer, Artilleriefeuer) **lokal** auf dem Server ausgeführt werden. Dadurch wird verhindert das **globale** Aufrufe mehrfach gestartet werden. Dies ist eine große Fehlerquelle in MP Spielen. Effekte dagegen werden nur für den Client sichtbar wenn diese **lokal** oder **global** auf der Rechenmaschine ausgeführt werden. Explosionen Schüsse dagegen sind immer **global** egal ob es sich hierbei um Client oder Server handelt.

Arbeiten mit Funktionen

Funktionen

Eines der letzten großen Geheimnisse in der ArMA Welt ist der Mythos Funktion. Diese werden im allgemeinen als sehr komplizierte geltenden Quellcodes angesehen, sind jedoch der Schlüssel zu vielen Problemen. Und anders als Skripte können Funktionen auch Werte liefern. Der Dateityp einer Funktion ist *.sqf. Dieser Typ kann in einem Editor einfach als Textform bearbeitet werden.

Funktion starten

Eine Funktion wird ähnlich wie ein Skript gestartet. Dies kann aus einer Init Zeile oder aus einem Skript oder aus einer Funktion erfolgen. Es können natürlich auch Parameter übergeben werden. Beispiel: [_object]execVM"TLAM\antrieb.sqf"; In diesem Fall wird die Funktion antrieb.sqf im Unterordner TLAM gestartet und ihr _object als Parameter mitgeliefert.

Anweisungen und Blöcke

Aus einem Ausdruck wie `_x = 1` oder hint "Alex hat grosse Fuesse" wird eine Anweisung wenn ihr ein Semikolon folgt. `_x = 1;` oder `hint "Alex hat grosse Fuesse";` Bei Funktionen in ArMA steht das Semikolon am Ende einer Anweisung. Die geschweiften Klammern { und } dienen dazu Anweisungen oder Vereinbarungen zusammen zu fassen. Diese Klammern werden auch für Anweisungen wie if-else oder while oder for Schleifen zum zusammenfassen verwendet. Die runden Klammern (und) dienen dazu Anweisungen oder Bedingungen zusammen zu fassen. Die Aufgabe `_a = 5 * 5 + 5;` Ergebnis `_a = 30` oder `_a = 5 * (5 + 5);` Ergebnis `_a = 50` können so beeinflusst werden. Desweiteren ist es manchmal nötig Funktionen Werte zu übergeben. `_a = sin (35 + 7);` Da sin einen Wert erwartet ist es von nöten diesen zuerst zu berechnen. Daher die Klammern.

Erklärungen hinterlassen

Um Erklärungen in Funktionen in Textform zu hinterlegen wird vor dem Text ein doppeltes // gesetzt. Somit kann jeder Befehl dokumentiert werden. `hint "Hallo"; //Hier werden der Aufruf gestartet Hallo aus zu geben.`

Pause

Um ein Skript anzuhalten besteht die Möglichkeit ~Wert zuschreiben. Das Skript würde jetzt eine Sekunde warten. In einer Funktion wird hierfür `sleep 1;` geschrieben.

Werte zuweisen

```
_a = 10;
```

weiß der lokalen Variable `_a` den Wert 10 zu.

Array einlesen auslesen

```
_a = [10,4,12];
```

Übergibt die Werte 10,4 und 12 an die lokale Variable `_a`.

```
_b = _a select 0;
```

```
_c = _a select 1;
```

```
_d = _a select 2;
```

ließt die Werte wieder aus `_a` und übergibt diese an `_b`, `_c` und `_d`. Das select 0 steht für die erste Position im Array. select 1 und select 2 für die zweite und dritte Position. Das selbe geht

natürlich auch mit globalen Variablen.

if-else

IF (Bedingung) Then {Aktion;};

Andere Schreibweise jedoch die Selbe Bedeutung.

IF (Bedingung) Then

```
{
  Aktion;
};
```

Wenn die Bedingung zutrifft, Aktion ausführen. Wenn diese Bedingung nicht zu trifft wird in der nächsten Zeile oder nach `};` weiter gemacht. Bedingungen werden in `()` Klammern zusammen gefast. Aktionen werden mit `{ }` Klammern zusammen gefast. Die Aktionen schließen wie bei jeder anderen auch mit einem `“ ; “` ab. Bedingungen nicht. Das Ende einer IF Kette wird mit `“ ; “` gesetzt.

Anwendungen verschachteln.

IF (Bedingung 1) Then

```
{
  IF (Bedingung 2) Then {Aktion 2;};
  IF (Bedingung 3) Then {Aktion 3;};
};
```

In diesem Fall werden die Bedingungen 2 und 3 nur abgefragt wenn die Bedingung 1 zutrifft. Dies kann bei einer Kette von Ereignissen sehr Hilfreich sein.

Entweder Oder

Eine weitere Variante der IF Abfrage ist die *IF (Bedingung) Then {Aktion a}; ELSE {Aktion b};* Abfrage. Hierbei wird eine Bedingung abgefragt, wenn diese zutrifft wird Aktion a ausgeführt. Trifft die Bedingung nicht zu wird Aktion b ausgeführt.

Schleifen

while

do

statement

while (expression);

Die bekannteste Schleife ist die while Schleife. *while {Bedingung} do {Aktion;};* Diese Schleife wiederholt sich solange bis die Bedingung nicht mehr zutrifft. Beispiel:

```
while {alive Alex} do           //Bedingung lebt Alex
{
  _a = GetDammage Alex;         //Schaden von Alex auslesen und in _a speichern
  Alex SetDamage (_a +0.1);     //Schaden + 0.1 dann den Schaden auf Alex übertragen
  sleep 1;                      //Pause für 1 Sekunde
};
```

In diesem Fall wiederholt sich die Schleife so lange bis die Einheit mit dem Namen Ales tot ist.

for

```
for (expression1; expression2; expression3)
    statement
```

äquivalent zu

```
expression1;
while (expression2) {
    statement
    expression3;
}
```

for zu bevorzugen wenn es um eine einfache Initialisierung geht, den for fast die kontrollierenden Anweisungen der Schleife zusammen und macht sie zu Beginn der Schleife sichtbar. Bedingungen werden nicht mit Semikolon abgeschlossen.

```
for [{_i = 0},{_i < 100},{_i = _i + 1}] do {Aktion};
```

`_i = 0` setzt den Zähler auf 0.

`_i < 100` Bedingung für die Ausführung

`_i = _i + 1` wird bei jedem Durchlauf ausgeführt.

Beispiel:

```
for [{_i = 0},{_i < 100},{_i = _i + 1}] do //durchläuft die Schleife 99 mal
```

```
{
    _a = GetDammage Alex;           //Schaden von Alex auslesen und in _a speichern
    Alex SetDamage (_a + 0.1);     //Schaden + 0.1 dann den Schaden auf Alex übertragen
    sleep 1;                       //Pause für 1 Sekunde
};
```

if-else und Schleifen verschachteln

```
if (expression)
    statement1
```

Zusammen geben diese Schleifen, Anweisungen und Zuweisungen dem Programmierer ein sehr mächtiges Werkzeug in die Hand.

```
IF (alive Sworn)Then
```

```
{
    IF (alive Alex) Then {
        for [{_i = 0},{_i <= 100},{_i = _i + 1}] do
        {
            hint format ["Zahl = %1",_i];
            sleep 1;
        };
    };
};
```

```
IF (alive Spider)Then
```

```
{
    while {alive Alex} do
    {
        _a = GetDammage Alex;
```

```

    Alex SetDamage (_a +0.1);
    sleep 1;
};
};

```

Sieht schlimmer aus als es ist. **Fragt ab ob Sworn lebt.** Wenn ja **Fragt ab ob Alex lebt.** Wenn ja **zählt bis 100 und gibt Werte aus.** Danach **Fragt ob Spider lebt.** Wenn ja **Bleibt in der Schleife bis Alex tot ist.**

Schleifen verschachteln

Was mit if-else läuft geht auch mit Schleifen unter einander.

IF (alive Sworn)Then

```

{
  while {alive Alex} do
  {
    for [{_i = 0},{_i <= 100},{_i = _i + 1}] do
    {
      hint format ["Zahl = %1",_i];
      sleep 1;
    };
  };
};

```

Fragt ab ob Sworn lebt. Wenn ja **Startet wenn Alex lebt** die **Schleife und zählt bis 100** und beginnt dann erneut wenn **wenn Alex lebt** mit der **Zähler Schleife und zählt bis 100.** Usw.

Wichtig ist bei solchen Strukturen das die Formatierung beibehalten wird. Es empfiehlt sich immer drei Zeichen einzurücken.

Das Fallschirmspringer Skript als Funktion

Es werden zwei Parameter übergeben. Name des Anführers der Gruppe und der Name des Vehicle aus dem die Einheiten Abspringen sollen. Zum ausführen **null=[GRP1,Air1]execVM "fallschirm.sqf";** aus einer Init Zeile, Skript, Funktion, Wegmarke oder Auslöser aufrufen.

```

//Arma Fallschirm Sprung Funktion Parameter Name der gruppe Name der Air Einheit
_gruppe = _this select 0;
_luft = _this select 1;
for [{_i = 0},{_i <= count units _gruppe},{_i = _i + 1}] do
{
  (units _gruppe select _i) action ["EJECT", _luft];
  unassignvehicle (units _gruppe select _i);
  sleep 1;
};

```

Die Fallschirmspringergruppe kann am besten über **{_x moveincargo Air1} foreach units GRP1;** in das Flugzeug gesetzt werden. GRP1 ist der Name des Anführers der Springer, Air1 der Name der Lufteinheit.

Das GPS Skript als Funktion

Es werden zwei Parameter übergeben. Name der Person und Name des Markers. Zum ausführen

`null=[Alex,Marker1]execVM "GPS.sqf";` aus einer Init Zeile, Skript, Funktion, Wegmarke oder Auslöser aufrufen.

//Arma GPS Funktion Parameter Name der Person Name des Markers

```
_einheit = _this select 0
_marker = _this select 1
while {alive _einheit} do
{
    "_marker " setMarkerPos getpos _einheit;
    sleep 2;
};
```

Wettersystem

Ich habe mich immer daran gestört das Helikopter und Flugzeuge keine Reaktion auf die Wettereinstellung zeigen. Daher dachte ich mir es wäre nett wenn man das mal ändert. Und da ich noch ein Beispiel für das Zusammenspiel von Skript und Funktion brauche passt mir das ganz gut. Zuerst werden wir ein Skript schreiben welches die Windrichtung und die Geschwindigkeit erfasst und in globalen Variablen 5-mal in der Sekunde heraus gibt. Dieses Skript starten wir mit den Parametern `[Name der Lufteinheit]exec "wetterdaten.sqs";` Quellcode `"wetterdaten.sqs"`:

```
_unit = _this select 0;
[_unit]execVM "wind.sqf"
#marker
xwind = wind select 0;
ywind = wind select 1;
~0.2
goto "marker"
```

Dieses Skript liest jetzt die Werte der Funktion Wind aus. In der globalen Variable `xwind` steht der aktuelle Wert der Luftbewegung auf der x Achse, in der globalen Variable `ywind` steht der aktuelle Wert der Luftbewegung auf der y Achse. Die Aktualisierung erfolgt fünfmal in der Sekunde. Beim Starten des Skriptes erhält dieses den Namen der Lufteinheit. Diese wird dann gleich an die Funktion `wetter.sqf` weiter übergeben. Diese Funktion wird mit dem Parameter `_object` gestartet. `_object` ist also der Name der Lufteinheit.

Die Aufgabe der `wind.sqf` ist es die Abweichung `xwind` und `ywind` auf den Helikopter ständig zu übertragen. Das muss die Funktion so lange machen bis die Lufteinheit zerstört ist. Und dann ist noch zu beachten dass die Einflüsse nur dann auf die Einheit wirken wenn diese nicht auf dem Boden steht. Die Schleife wird mit der `while {lebt Einheit} do {Werte an Objekt übertragen wenn das Objekt in der Luft};` Hört sich schlimmer an als es ist. Weitergehend kann die Geschwindigkeit nicht einfach überschrieben werden, da die Einheit sonst beliebig beschleunigt würde. Daher werden die `xwind` und `ywind` Werte mit einem Faktor multipliziert. Ist keine perfekte Lösung hat sich jedoch als nützlich erwiesen. Die Übertragung erfolgt jetzt
Quellcode `wind.sqf`:

```
_unit = _this select 0;
```

```

while {alive _unit} do
{
    _x = 0.1 * xwind;
    _y = 0.1 * ywwind;
    // sensor fliegt die Einheit?
    _zreal = getPos _unit select 2;
    // Wind an Einheit übergeben.
    if (_zreal > 0) then {_unit setvelocity [(velocity _unit select 0) + _x ,(velocity _unit
select 1) + _y,(velocity _unit select 2)];};
    sleep 0.2;
};

```

Das war es schon. Kleine Funktion große Wirkung. Und das Fliegen im Sturm wird zu einem Tanz auf dem Drahtseil. Dieses Beispiel ist jedoch nur eine Variante. Der Programmierer kann noch eine Abfrage für das Gelände mit einbinden um Auf- und Ab- Winde zu realisieren.



Quellcode:

```

_unit = _this select 0;
_sensor = "HeliHEmpty" createVehicle [0,0,0];

while {alive _unit} do
{
    _wspeed = sqrt (xwind * xwind + ywwind * ywwind);
    _x = xwind / _wspeed;
    _y = ywwind / _wspeed;
    // sensor Boden?
    _sensor setPos [(getPos _unit select 0) + _x,(getPos _unit select 1) + _y];
    _poss1 = getPosASl _sensor select 2;
    _sensor setPos [(getPos _unit select 0) - _x,(getPos _unit select 1) - _y];
    _poss2 = getPosASl _sensor select 2;

    _berg = (_poss1 - _poss2)/4;

    _zreal = getPos _unit select 2;
    // Wind an Einheit übergeben.
    if (_zreal > 0.1) then {_unit setvelocity [(velocity _unit select 0) ,(velocity _unit select
1),(velocity _unit select 2)+ _berg ]};};
    sleep 0.2;
};

```

Diese Methoden sind keine genaue Abbildung von Wind und Sturm. Sie ist ein Beispiel für das Zusammenspiel von Funktion und Skript sowie von globalen und lokalen Variablen.

Windsystem in einer Funktion. Es werden neben der Abweichung durch Wind, sowie den Aufwinden an Bergen auch Effekte des Rotors im Tiefflug beachtet.

Quellcode windsystem.sqf:

```

_object = _this select 0;

//Erstellt den sensor für die Erfassung des Gelände
_sensor = "HeliHEmpty" createVehicle [0,0,0];

//Zeiteinstellung für die Tacktrate der Funktion
_t = 0.05;

while {alive _object} do
{
    // Frage ob _object fliegt oder am Boden steht.
    _a = (getPos _object select 2);

    // Wenn _object fliegt ausführen
    if (_a > 0.1) then {

        // Rotoreffect vom Boden. Flughöhe und Geschwindigkeit
        _z = -0.075 * _a + 0.4;
        if (_z < 0) then {
            _z = 0
        };
        _b = speed _object;
        _zb = -0.004 * _b + _z;
        if (_zb < 0.1) then {
            _zb = 0
        };

        //erfassen der Windrichtung in Arma
        _xwind = wind select 0;
        _ywind = wind select 1;

        //Auf und Abwind erfassen 1m in Windrichtung
        _wspeed = sqrt (_xwind * _xwind + _ywind * _ywind);

        _xs = _xwind / _wspeed;
        _ys = _ywind / _wspeed;
        _sensor setPos [(getPos _object select 0) + _xs,(getPos _object select 1) + _ys];
        _poss1 = getPosASI _sensor select 2;
        _sensor setPos [(getPos _object select 0) - _xs,(getPos _object select 1) - _ys];
        _poss2 = getPosASI _sensor select 2;
        _berg = (_poss1 - _poss2);

        //Berechnen von auf und ab sowie Abdrift
        _zb = _zb + (_wspeed * _berg / 2 * _t);
    }
}

```

```

//Windeinfluss auf _object
_x = _t * _xwind ;
_y = _t * _ywind ;

//Übergabe an das _object
_object setVelocity [(velocity _object select 0) + _x,(velocity _object select 1)
+ _y,(velocity _object select 2) + _zb];
};

sleep _t;
};

```

Funktionen entwickeln

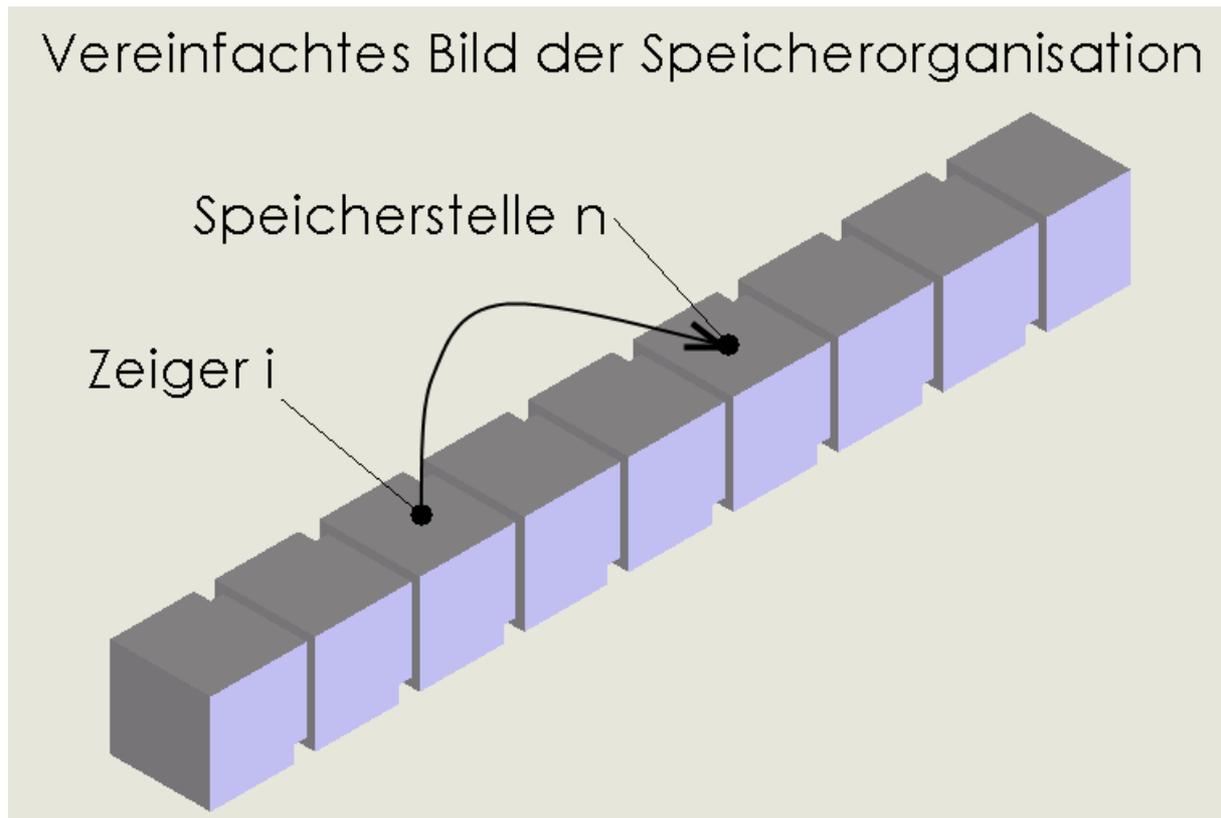
In diesem Kapitel wird experimentell beschrieben wie eine Funktion zu entwickeln ist. Wir haben in den bisherigen Kapiteln Editorbeispiele sowie Skripte und Funktionen kennengelernt. In diesen Skripten wurden Variablen und vor definierte Befehle verwendet. Was sind diese Befehle eigentlich? Arma hat einige Hundert dieser Befehle, welche vom Programmierer frei zu nutzen sind. Jedoch bleiben viele Fragen ungeklärt.

Zeiger und Vektoren

Ein Zeiger ist nichts anderes als eine Variable. Jedoch enthält dieser keinen Wert, sondern eine Adresse. Wird dieser Zeiger angesprochen zeigt dieser auf deine andere Variable und liefert den Wert der Variablen. Ein Zeiger zeigt auf eine Speicherstelle in der Maschine. Bei einer üblichen Architektur kann jedes Byte ein char-Wert sein, ein Paar Byte Zellen ein short, und vier aneinander liegende Byte ein long.

Zeiger und Adressen

Vereinfachtest Bild der Speicherorganisation. Eine normale Maschine hat eine fortlaufende Anzahl von Speicherzellen, welche nummeriert sind. Diese können alleine oder als Gruppe behandelt werden, und angesprochen werden. In diesem Bild gibt es einen Zeiger (i genannt) und eine Speicherzelle (n genannt).



In diesem Beispiel weiß der Zeiger `i` auf eine Speicherbank `n` zu. Das heißt, `i` zeigt auf `n`. Dadurch wird es möglich an Funktionen ganze Zeiger für Variablen zu übergeben, so dass diese dann mit denen rechnen / arbeiten können.

Zeiger und Funktionsargumente

In diesem Beispiel wollen wir eine Funktion, ich nenne diese POS Tausch untersuchen. Ihr Ziel ist es zwei Werte aus zu tauschen. (Typ der werte Ganzzahl, kein Return)

Quellcode mit Argumenten:

```
void tausch (int x, int y) //Fehler//
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}
```

Da die Beiden Elemente `x` und `y` als Argumente übergeben worden kann `tausch` diese nur in der Funktion verändern. Einen Einfluß hat diese Funktion nicht.

Quellcode mit Zeigern:

```
void tausch (int *x, int *y) //Richtig//
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}
```

gestartet wird diese Funktion mit `tausch (&x, &y)`. Nun werden die Werte nicht als Argumente sondern als Zeiger auf die Speicher Stellen übergeben.

Distanz XY und XYZ

Diese Funktion ist das Gegenstück zu der BI Funktion `diszanz`.
 Beispiel Einheit1 Distanz Einheit2 Return Entfernung in Metern.
 Quellcode `distanz`:

```
float distanz (char einheit1[], char einheit2[])
{
    float x1, x2, y1, y2, d;
    x1 = getpos einheit1[0];
    x2 = getpos einheit2[0];
    y1 = getpos einheit1[1];
    y2 = getpos einheit2[1];
    d = sqrt ((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
    return d;
}
```

Das gelieferte Ergebnis ist in diesem Fall die Entfernung beider Einheiten, berechnet in einem 2d Raum X/Y. Diese Funktion liefert jedoch nur den Richtigen Wert, der Entfernung, wenn beide Einheiten auf der gleichen Höhe stehen.

Quellcode `distanz 2`:

```
float distanz (char einheit1[], char einheit2[])
{
    float x1, x2, y1, y2, z1, z2, d;
    x1 = getpos einheit1[0];
    x2 = getpos einheit2[0];
    y1 = getpos einheit1[1];
    y2 = getpos einheit2[1];
    z1 = getpos einheit1[2];
    z2 = getpos einheit2[2];
    d = sqrt ((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2) + (z1 - z2) * (z1 - z2));
    return d;
}
```

Diese Funktion ist mit der ersten gleich. Sie berücksichtigt jedoch auch den Höhenunterschied.

Erste eigene Funktion

Diese Funktion soll die x, y und z Koordinate von zwei Einheiten tauschen. Dies kann zwar auch über getPos Einheit 1 und setPos Einheit 2 realisiert werden jedoch wollen wir das ohne vordefinierte Befehle machen. (experimentelle Idee. Es wird gearbeitet.)

{Im Moment ist das hier noch eine Baustelle}



Effekte

Feuer in der Mission

Es gibt im Editor unter Objekte sogenannte Feuerstellen. Diese können durch den Aufruf **this inflame true**; zum brennen gebracht werden oder durch **this inflame false** gelöscht werden. Eine schöne Spielerei ist es, diese Feuer in Schornsteinen zu platzieren. Dies wirkt dann als, ob diese rauchen würden

Rauchgranaten und Flare erzeugen

Um eine Rauchgranate zu erzeugen gibt es verschiedene Wege. Der einfache Weg ist diese einfach zu erzeugen, ohne auf eine Einheit zu achten. Diese entsteht dann wie aus dem nichts. Dazu erstellt man im Editor einfach ein Objekt. Es empfiehlt sich der Heliport (empty) da dieser nicht sichtbar ist und keine Geometrie hat. Über den Befehl:

```
Rauchgranate1="SmokeShellGreen" CreateVehicle [(getPos Heliport1 select 0),(getPos Heliport1 select 1),23]
```

in einem Auslöser oder Skript oder einer Init Zeile kann man jetzt eine Rauchgranate, Flare oder jedes andere Objekt erzeugen. Die 23 ist hierbei die die Angabe der Höhe. Das Ganze funktioniert in etwa so. Das Skript holt sich die Position der Unit mit dem Namen Heliport1 liest die Höhe aus und beschreibt diese erneut mit 23 was für 23m steht. Die Höhe bezieht sich über die am Boden. Diese erscheinen dann jedoch wie gesagt aus Geisterhand.

Getestet mit: F_40mm_White, F_40mm_Green, F_40mm_Red, F_40mm_Yellow, G_40mm_HE, SmokeShellRed, SmokeShellGreen, SmokeShell, GrenadeHandTimed, GrenadeHand ;

Diese Möglichkeit gibt einer Mission etwas Ambiente. Über der Feindbase leuchtet eine Flare auf, Ein Helikopter landet im Rauch...

Die etwas realistischere Möglichkeit ist die das eine Einheit diese Effekte selber auslöst. Das heißt sie wirft Rauchgranaten, feuert Flares in den Nachthimmel. Voraussetzung hierfür ist jedoch das diese Einheit im Besitz dieser Waffen ist.

Zuerst müssen wir jedoch die Einheit mit den Notwendigen Gegenständen und Abschussvorrichtungen ausrüsten. Der einfachste Weg ist der diese zuerst zu entwaffnen.

Befehl in der Init Zeile der Einheit: **removeAllWeapons this**

jetzt kann dieser nach eigenem Ermessen bewaffnet werden. Dieser soll mit M16A2GL und 2rot 2gelb und 2 weißen Leuchtgranaten bestückt werden. Auszug aus der Initzeile der Einheit.

```
removeAllWeapons this;
this addMagazine "FlareRed_M203";
this addMagazine "FlareRed_M203";
this addMagazine "FlareYellow_M203";
this addMagazine "FlareYellow_M203";
this addMagazine "FlareWhite_M203";
this addMagazine "FlareWhite_M203";
this addWeapon "M16A2GL";
```

Nachdem die Einheit ausgestattet ist lassen wir diese in die Luft schießen.

```
Soldat1 fire ["M203Muzzle","M203Muzzle","FlareRed_M203"]
```

Der Soldat feuert nun eine rote Leuchtugel in den Himmel. Natürlich nur solange er diese Waffe besitzt und am Leben ist. Wenn diese Einheit aufgefordert wird eine Granate abzuschießen jedoch alle dieses Typs abgeschossen hat feuert diese einfach den nächsten Typ der Waffe ab. So kann es passieren, dass die Einheit eine gelbe anstelle einer roten Flare abfeuert.

Dies lässt sich auch auf andere Waffen anwenden. Dazu werden die Typen ausgetauscht.
Beispiel Ost Soldat:

```
removeAllWeapons this;
this addMagazine "FlareWhite_GP25";
this addMagazine "FlareWhite_GP25";
this addWeapon "AK74GL";
bewaffnet den Soldaten mit zwei weißen Flare.
Soldat2 fire ["GP25Muzzle","GP25Muzzle","FlareWhite_GP25"]
```

Sprengungen erzeugen auslösen

Um Sprengladungen von der AI legen zu lassen muss die Voraussetzung, dass diese die Bomben auch bei sich trägt, erfüllt sein. An einem Wegpunkt oder Auslöser kann über den Befehl: `Bombenleger1 fire ["pipebombmuzzle","pipebombmuzzle","pipebomb"]` der Sprengsatz gelegt werden. Bombenleger1 ist der Name der Einheit. Über den Befehl: `Bombenleger1 action ["TOUCHOFF",Bombenleger1]` wird dieser nun zur explosion gebracht.

Brennendes explodierendes Fass

Ein schöner Effekt kann auch mit den Benzinfässern geschaffen werden. Hierzu platziert ein Fass auf der Karte. Tragt in die Init Zeile des Fass [Name des Fass] exec "`explo.sqs`" ein. Erstellt in dem Missionsordner ein Skript mit dem Namen `explo,sqs`.
Quellcode:

```
_unit = _this select 0;
@ not alive _unit
~20 - random 5

Neu1 = "R_Hydra_HE" CreateVehicle getPos _unit;
_x = 1 - random 2
_y = 1 - random 2
_z = 15 - random 10
_unit setvelocity [_x,_y,_z]
exit
```

Das führt da zu, dass sobald das Fass kaputt ist und anfängt zu brennen, das Skript 15 bis 20 Sekunden wartet und dann an der Stelle des Fass eine Sprengladung erstellt. Diese explodiert und mit dem Befehl `_unit setvelocity [_x,_y,_z]` wird das Fass nach oben und zu den Seiten beschleunigt. Das wirkt dann so als ob das Fass explodiert und durch den Druck durch die Gegend fliegt. (Ein kleiner Hauch von Hollywood)



Feuer breitet sich aus

Stellt zwei Fässer (Fass1 und Fass2) neben einander. (Oder auch mehr) Erstellt einen Auslöser, welcher abfragt ob Fass1 nicht lebt. Sobald Fass1 nicht mehr lebt soll dieser Auslöser das Fass2 zerstören. Dazu Fass2 setdamage 1 einsetzen. Das ganze kann jetzt noch mit einer Zeit min 5 max 10 mid 7 versehen werden, damit das Feuer nicht sofort übergreift. Dies gibt eine schöne Kettenreaktion wenn das mit dem Skript von Brennendes explodierendes Fass kombiniert wird.



Partikel erzeugen (Drop Array)

Bei Partikeln handelt es sich um alle Effekte wie Rauch Feuer Wasser Staub etc. Diese Effekte können in verschiedene Richtungen gelenkt werden xyz dabei Farben und Form ändern und sind in Grundmodelle aufgeteilt die Struktur und Form regeln.

Namen der Partikel

```
"\Ca\Data\ParticleEffects\Blood\Blood.p3d",
"\Ca\Data\Cl_basic.p3d",
"\Ca\Data\Cl_fire",
"\Ca\Data\Cl_fired",
"\Ca\Data\Cl_water",
"\Ca\Data\ParticleEffects\CloudletSand\CloudletSand.p3d",
"\Ca\Data\ParticleEffects\CloudletWater\CloudletWater.p3d",
"\Ca\Data\ParticleEffects\FireAndSmokeAnim\FireAnim.p3d",
"\Ca\Data\kouleSvetlo",
"\Ca\Data\krater_po_kulce",
"\Ca\Data\missileSmoke",
"\Ca\Data\ParticleEffects\Pstone\Pstone.p3d",
"\Ca\Data\RainDrop",
"\Ca\Data\ParticleEffects\RocketSmoke\RocketSmoke.p3d",
"\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
"\Ca\Data\ParticleEffects\SparksEffect\SparksEffect.p3d",
"\Ca\Data\ParticleEffects\Watereffects\WaterEffects.p3d"
```

```
[["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d", 8, 5, 1], "", "Billboard",
1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7, 0.4], [0.5, 0.5, 0.5, 0.2], [0.2,
0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Das sieht auf den ersten Blick recht kompliziert aus. Ist es jedoch nicht. Es handelt sich lediglich um eine Funktion welche mit Parametern gestartet wird. Das sehen wir uns jetzt etwas genauer an.

```
[["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d", 8, 5, 1], "", "Billboard",
1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7, 0.4], [0.5, 0.5, 0.5, 0.2], [0.2,
0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Grundlagen Partikel

Erzeugen eines Partikel (Falsche Unmögliche Einstellungen haben einen Absturz zur Folge)

Typ eines Partikel

Es gibt zwei Grundmodelle für Partikel "Billboard" und "SpaceObject".



Links Billboard

rechts SpaceObject

Es wird hier die Struktur des Objekt bestimmt. Die Einstellung des Effektes ist ansonsten identisch. Sworn ist der Name des Particels.

```
Sworn setParticleParams [["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7, 0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Farben eines Partikel

Die Farbe eines Objectes ist in drei Grundfarben und einem Wert für die Gesamtintensität aufgeteilt. Aus der Mischung und Stärke der einzelnen Farben wird der Farbton generiert.

Particle_Color Farbton **rot**

Particle_Color Farbton **grün**

Particle_Color Farbton **blau**

Particle_Color Farbstärke [[Startfarbe][Übergangsfarbefarbe][Endfarbe]]

Particle_Color Farbstärke [[**rot,grün,blau**,Stärke],[**rot,grün,blau**,Stärke],[**rot,grün,blau**,Stärke]]

Eine weitere Besonderheit ist das der Effekt eine Startfarbe eine Übergangsfarbe und eine Endfarbe hat. Daher ist es möglich einen Effekt verblassen zu lassen bevor endet. Die Werte sind in ihrer Größe nicht begrenzt. Lediglich die Mischung der Grundfarben ist ausschlagsgebend. Ab version 0.5 ist es möglich mehr Farben zu bestimmen. Diese werden dann der Reihe nach aufgeführt.

[[Farbe 1],[Farbe 2],[Farbe 3],[...],[...],[...],[Farbe N]] die zuordnung
[**rot,grün,blau**,Stärke] ändert sich nicht.

Startfarbe

```
Sworn setParticleParams [["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7, 0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Übergangsfarbefarbe

```
Sworn setParticleParams [["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7, 0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Endfarbe

```
Sworn setParticleParams [{"Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7 ,0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Es ist nicht zwingend nötig nur drei Farben zu definieren. Es können auch mehr sein. Drei Farben haben sich jedoch als ausreichend erwiesen.

Größe eines Partikel

Hier wird die Größe des einzelnen Partikels eingestellt. Breite und Höhe

```
Sworn setParticleParams [{"Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7 ,0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Gewicht eines Partikel

Das Gewicht regelt das Verhalten eines Partikel gegenüber der Schwerkraft. Der gewählte Wert (Typ float) regelt das steigen oder sinken. Werte größer 1.5 lassen das Partikel sinken kleinere steigen.

```
Sworn setParticleParams [{"Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7 ,0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Volumen eines Partikel

Regelt die Beschleunigung eines Partikel nach oben und unten.

```
Sworn setParticleParams [{"Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7 ,0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Drehung eines Partikel

Eigene Rotation des Partikels

```
Sworn setParticleParams [{"Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7 ,0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Lebenszeit eines Partikel

Zeit bis zum Löschen in sec.

```
[["Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d", 8, 5, 1], "", "Billboard",
1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7 ,0.4], [0.5, 0.5, 0.5, 0.2], [0.2,
0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Position eines Partikel vom Objekt

Hier wird die Position festgelegt in welcher das partikel vpm Objekt (hier Objekt Name Alex) erstellt wird. [X,Y,Z]

```
Sworn [{"Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d", 8, 5, 1], "",
"Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7 ,0.4], [0.5, 0.5, 0.5,
0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Alex];
```

Grundeinstellungen Partikel

```
particleSource setParticleCircle [radius, velocity]
particleSource setParticleParams [array]
```

particleSource setParticleRandom [lifeTime, position, moveVelocity, rotationVelocity, size, color, randomDirectionPeriod, randomDirectionIntensity]

```
setParticleParams [["Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d", 8, 5,
1], "", "Billboard", 1, Lebenszeit, [rel pos], [0, 0, 0.5], Drehung, Gewicht, Volumen, 0.1,
[Größe], [[Startfarbe], [Übergangsfarbe], [Endfarbe]], [0, 1], 1, 0, "", "", "_b];
```

Lebenszeit:

Hier wird die Lebenszeit \pm der Voreinstellungen berechnet.

Sworn **setParticleRandom** [0, [0, 0, 0], [0, 0, 0], 0, 0, [0, 0, 0, 0], 0, 0];

Position

Bestimmung des Startpunktes. [XYZ] (Werte werden \pm mit den eingestellten Größen gerechnet)

Sworn **setParticleRandom** [0, [0, 0, 0], [0, 0, 0], 0, 0, [0, 0, 0, 0], 0, 0];

move Velocity

Diese Zufallseinstellung regelt die Geschwindigkeit XYZ mit der eine Abweichung zur Normalen Geschwindigkeit erzielt wird. (Werte werden \pm mit den Eingestellten Größen gerechnet)

Sworn **setParticleRandom** [0, [0, 0, 0], [0, 0, 0], 0, 0, [0, 0, 0, 0], 0, 0];

Color

Die Farbe [rot,grün,blau,Stärke] wird hier eingestellt. (Werte werden \pm mit den Eingestellten Größen gerechnet)

Size Größe des Partikel (Zufalls Größe des Partikel) (Werte werden \pm mit den Eingestellten Größen gerechnet)

Sworn **setParticleRandom** [0, [0, 0, 0], [0, 0, 0], 0, 0, [0, 0, 0, 0], 0, 0];

setDropInterval interval

Hier wird dem user die Möglichkeit gegeben die Wiederholung eines Partikel zu regeln. (Wahrte Zeit zur erstellung eines neuen in Sekunden)

Sworn **setDropInterval** 0.04;

Es schneit

Es gibt in ArMA eine eingebundene Funktion mit der Rauch und Feuer Effekte erzeugt werden können. Diese sind die aus OFP bekannten Drop Arrays. So ist immer noch keine Erklärung, jedoch ein Beispiel.

```
i=0
#loop
_pos = [ (position player select 0)+random 50,(position player select 1)+random
50,20+random 50 ];
drop ["ca\data\cl_basic", "", "Billboard", 0.01, 100, _pos, wind, random 1, 1+random
.5, 1, 0.2, [random 1], [[1, 1, 1], [1, 1, 1, 0.7]], [1, 0], 1, 1, "", "", ""];
~.1
?(i<1000):goto "loop"
```

Dieser Aufruf erzeugt ein Partikel, Farbe weiß, welches eine Masse hat und über eine Zufalls Position erzeugt wird. Das ganze wiederholt sich 1000 mal. (es schneit. LOL). Dieses Thema würde alleine eine Anleitung dieser Größe füllen. Hier noch einige Grundlagen.

Erstellen einer Rauchsäule

Hier ist ein Beispiel zur Erstellung einer Rauchsäule gezeigt. Hierbei handelt es sich um ein Skript welches eine Anzahl von Partikeln erzeugt welche aufsteigen. Dadurch entsteht der Effekt dass es um eine Rauchsäule handelt. Zuerst erstellen wir ein Vehicle mit dem Namen Luft1. (Das dient der Positionsbestimmung) und starten in dessen Initzeile den Aufruf für ein Skript. `[exec "smoke.sqs"`

Beispiel Skript smoke.sqs:

```
_rauch = "#particlesource" createVehicleLocal getpos Luft1;
_rauch setParticleCircle [0, [0, 0, 0]];
_rauch setParticleRandom [0, [2.5, 2.5, 0], [0.75, 0.75, 0], 0, 0, [0.05, 0.05, 0.05, 0.05], 0, 0];
_rauch setParticleParams [["Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7, 0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Luft1];
_rauch setDropInterval 0.04;
```

In diesem Beispiel wurde jetzt `_rauch` als lokale Variable erstellt,

```
_rauch = "#particlesource" createVehicleLocal getpos Luft1;
```

und die Position von Vehicle Luft1 übergeben.

```
_rauch setParticleCircle [0, [0, 0, 0]];
```

Hier kann ein Radius angegeben werden um den die Partikel erstellt werden sollen, sowie die Geschwindigkeit in der dieser Radius umkreist wird.

```
_rauch setParticleRandom [0, [2.5, 2.5, 0], [0.75, 0.75, 0], 0, 0, [0.05, 0.05, 0.05, 0.05], 0, 0];
```

Der Random Aufruf übergibt eine Reihe von Werten. (Erklärung Grundlagen Partikel)

```
_rauch setParticleParams [["Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7, 0.4],
[0.5, 0.5, 0.5, 0.2], [0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", Luft1];
```

Das Hauptarray enthält alle Werte (Farben Gewicht etc. Erklärung Grundlagen Partikel).

```
_rauch setDropInterval 0.04;
```

Und zuletzt die Angabe des Intervalls. (in Sekunden) (es ist besser dieses mit der Framerate des Systems zu verknüpfen)

Noch eine Rauchsäule

Erstellt im Editor ein unsichtbares H. Nennt es zB. Smoke1. Nun platziert ihr das H auf die Position eines Kamines an einem Haus. Siehe hierzu Kapitel Editieren – Objekte/Einheiten schweben lassen. Nun macht ein Auslöser. Aktivierung Spielerseite. In die Init des Auslösers schreibt ihr `[smoke1] exec "chimney.sqs"`;

Folgendes in die chimney.sqs kopieren und im Spieleordner abspeichern.

```
; creates smoke for smoking chimneys. to use this script:
; create a object (eg. heliport empty). Name it eg, smoke1
; create a trigger: [smoke1] exec "chimney.sqs"
; place the object in height of the chimney
```

```
_b = _this select 0
```

```
_rauch = "#particlesource" createVehicleLocal getpos _b;
```

```
; _rauch setParticleCircle [radius, [velocity]]
```

```
_rauch setParticleCircle [0, [0, 0, 0]];
```

```

;_rauch setParticleRandom [lifeTime,[position],[moveVelocity], rotationVelocity, size,
;[color],          ;randomDirectionPeriod, randomDirectionIntensity]
_rauch setParticleRandom [0, [0.5, 0.5, 0], [0, 0, 0], 0, 0.2,[0.05, 0.05, 0.05, 0.05], 0, 0];

;_rauch setParticleParams
;[["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d", 8, 5, 1], "", ;"Billboard",
;1, Lebenszeit, [rel pos], [0, 0, 0.5], Drehung, Gewicht, Volumen, 0.1, [Größe],[[Startfarbe],
;[Übergangsfarbe],[Endfarbe]],[0, 1], 1, 0, "", "", _b];
_rauch setParticleParams [["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 5, 1], "", "Billboard", 1, 4,[0, 0, 0], [0, 0, 0.5], 0, 0.6, 0.5, 0.1,[0.6, 0.6],[[0.7, 0.7, 0.7
,0.4],[0.5, 0.5, 0.5, 0.2],[0.2, 0.2, 0.2, 0]], [0, 1], 1, 0, "", "", _b];

_rauch setDropInterval 0.04;
exit

```

Bei vielen Missionen fehlt Leben, das besondere Extra. Rauchende Kamine sind da eine gute Abhilfe.



Erstellen einer Feuer Rauchsäule

Die Vorbereitungen sollten wie bei der Rauchsäule getroffen werden. Den Skripteintrag ändern, damit Feuer und Rauch erstellt werden können. Es wird hierzu ein Objekt erstellt. name Haus. Über einen Skriptaufruf wird das Feuer jetzt gestartet. `[Haus] exec "brennen.sqs"`



```

_objekt = _this select 0

_feuer = "#particlesource" createVehicleLocal getpos _objekt;
_feuer setParticleCircle [1, [1, 4, 1]];
_feuer setParticleRandom [0.2, [1, 1, 0], [2, 2, 1], 0.2, 0.2, [0, 0, 0, 0], 0, 0];
_feuer setDropInterval 0.05;
_feuer setParticleParams [["\ca\data\ParticleEffects\FireAndSmokeAnim\FireAnim", 8, 2, 7],
"", "Billboard", 1, 1, [random 0.5, random 0.5, 0], [0, 0, 2], 1, 1, 0.9, 0.3,
[4,6],[[1,1,1,0.7],[1,1,1,0.5],[1,1,1,0]], [0,1], 1, 1, "", "", _objekt]

_feuer1 = "#particlesource" createVehicleLocal getpos _objekt;
_feuer1 setParticleCircle [1, [0.5, 3, 0]];
_feuer1 setParticleRandom [0.5, [1, 1, 0.4], [0, 0, 4], 0, 0.5, [0, 0, 0, 0], 0, 0];
_feuer1 setDropInterval 0.01;
_feuer1 setParticleParams [["\ca\data\ParticleEffects\FireAndSmokeAnim\FireAnim", 8, 2,
1], "", "Billboard", 1, 2, [0, random 0.5, random 1], [0, 0, 1], 10, 1, 0.9, 0.3,
[1],[[1,1,1,0.5],[1,1,1,0.2],[1,1,1,0]], [0.5,0.5,0], 1, 1, "", "", _objekt];

_rauch1 = "#particlesource" createVehicleLocal getpos _objekt;
_rauch1 setParticleCircle [0, [0, 0, 0]];
_rauch1 setParticleRandom [0, [2.5, 2.5, 0], [0.75, 0.75, 0], 0, 0, [0.05, 0.05, 0.05, 0.05], 0, 0];
_rauch1 setParticleParams
[["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d", 8, 5, 1], "", "Billboard",
1, 13, [0, 0, 0], [0, 0, 0.5], 1, 1.2, 1, 0.1, [5, 5], [[0.7, 0.7, 0.7 ,0.4], [0.5, 0.5, 0.5, 0.2], [0.2,
0.2, 0.2, 0]], [0, 1], 1, 0, "", "", _objekt];
_rauch1 setDropInterval 0.04

```

Feuer muss bei Nacht leuchten!

Desweiteren empfiehlt es sich bei Feuer ein Licht mit ein zu bauen. Denn was ist schon Feuer ohne Licht. Hierzu wird ein weiteres Skript aufgerufen. Diesem werden dann etliche Parameter übergeben.

```
[Haus,0,0,0.1,0.2,0.1,0.1,0.1,0.02,0.01,0.01] exec "light.sqs"
```

Quellcode:

```

_objekt = _this select 0
_x = _this select 1
_y = _this select 2
_z = _this select 3
_rot = _this select 4
_gruen = _this select 5
_blau = _this select 6
_hell = _this select 7
_rotu = _this select 8
_gruenu = _this select 9
_blauu = _this select 10
_light = "#lightpoint" createVehicleLocal [0,0,0];
_light setLightBrightness _hell;
_light setLightAmbient[_rotu, _gruenu, _blauu];
_light setLightColor[_rot, _gruen, _blau];
_light lightAttachObject [_objekt, [_x,_y,_z]]

```

siehe zu diesem Thema bitte Erstellen von Lichtquellen.

Kamin in Aktion

Um die Kamine einer Fabrik zum rauchen zu bringen und damit das Umfeld lebendig erscheinen zu lassen, setzen wir einen Marker oder eine Logik an den Punkt [x,y] an dem der Rauch erstellt werden soll. Diesen nennt ihr Rauch1. In die Initzeile startet jetzt eine Funktion oder ein Skript.

```
Skript[Rauch1]exec "arbeit1.sqs"
Funktion null=[Rauch1]execVM"arbeit1.sqf"
in dieser werden dann die Werte aufgerufen.
Quellcode Skript
```



```
_rauch = "#particlesource" createVehicleLocal getpos Rauch1;
_rauch setParticleRandom [2, [0.3, 0.3, 0], [0.25, 0.25, 0.25], 1, 0.5, [0.1, 0.1, 0.1, 0.1], 1, 0];
_rauch setParticleParams [["Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 1, 8], "", "Billboard", 1, 15, [0, 0, 30], [0, 0, 0.75], 0, 10.6, 8.5, 0.075, [1.2, 2, 4], [[0.1, 0.1,
0.1, 1], [0.15, 0.15, 0.15, 0.7], [0.2, 0.2, 0.2, 0.5], [0.3, 0.3, 0.3, 0]], [0.08], 1, 0, "", "",
Rauch1];
_rauch setDropInterval 0.04;
```

Wer hier aufmerksam liest wird sich fragen warum hier 4 anstelle von 3 Farben definiert sind. Das ist eine der Verbesserungen welche in Version v 0.5 von ArMA zustande kommt. Es gibt keine Grenzen in der Farbanzahl.

Feuer im Triebwerk



In diesem Beispiel wird dem linken Triebwerk das Rauchen beigebracht. Es können auch weitere Effekte eingebunden werden wie Feuer oder Licht. Jedoch ist das immer dasselbe. Weitergehend wird der Effekt nach der Erstellung noch weiter verändert. Der erste Teil des Skriptes legt den Typ fest, die Position der Partikel am Heli, und die Richtung und dessen Geschwindigkeit. In der Schleife wird dann diese Geschwindigkeit verändert und an die Bewegung angepasst. Dies soll den Einfluss des Rotors simulieren. Starten des Skript mit dem Parameter Name der Einheit.

Quellcode:

```

_unit = _this select 0;
_rauch = "#particlesource" createVehicleLocal getpos _unit;
_rauch setParticleRandom [2, [0.3, 0.3, 0], [0.25, 0.25, 0.25], 1, 0.5, [0.1, 0.1, 0.1, 0.1], 1, 0];
_rauch setParticleParams [["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 1, 8], "", "Billboard", 1, 15, [-1, -1, 0], [0, 0, -1], 0, 10.4, 8.5, 0.075, [1.2, 2, 4], [[0.1, 0.1,
0.1, 1], [0.15, 0.15, 0.15, 0.7], [0.2, 0.2, 0.2, 0.5], [0.3, 0.3, 0.3, 0]], [0.08], 1, 0, "", "", _unit];
_rauch setDropInterval 0.04;

#marke
_a = 10 / (ABS (speed _unit) + 1)
_rauch setParticleParams [["\Ca\Data\ParticleEffects\FireAndSmokeAnim\SmokeAnim.p3d",
8, 1, 8], "", "Billboard", 1, 15, [-1, -1, 0], [0, 0, -_a], 0, 10.4, 8.5, 0.075, [1.2, 2, 4], [[0.1, 0.1,
0.1, 1], [0.15, 0.15, 0.15, 0.7], [0.2, 0.2, 0.2, 0.5], [0.3, 0.3, 0.3, 0]], [0.08], 1, 0, "", "", _unit];
~0.1
?alive _unit: goto "marke"
exit

```

Um zu verhindern das das Triebwerk die ganze Zeit raucht, kann man @ dammage _unit > 0.3 zu Beginn einfügen. Jetzt wird erst ab einem Schaden von 0.3 Rauch erzeugt.

Erstellung von Lichtquellen

Mit den folgenden Befehlen lassen sich Lichtquellen erzeugen um zB ein Dorf mit Lichtquellen auszustatten. Dies kann zur „Belebung“ eines Szenarios schon viel beitragen.

Erste Methode zur Erstellung einer Lichtquelle:

1. Man setze im Editor ein unsichtbares H und benenne es mit Licht
2. Man setze eine Spielfigur und schreibe in die init: **this exec "light.sqs"**. (Die Auslösung des Scriptes kann natürlich auch durch einen Trigger oder durch Eintrag in der init.sqs erfolgen)
3. Man erstelle ein Script mit dem Microsoft Editor (Programme – Zubehör – Editor) und nenne es light. Fügt folgendes ein und geht dann auf Speichern unter. Nennt es light.sqs.
4. Die light.sqs muss natürlich in eurem Missionsordner sein um mit den anderen Scripten geladen zu werden

Beispiel einer weisen Lichtquelle nach der ersten Methode:

```

_light = "#lightpoint" createVehicleLocal [0,0,0];
_light setLightBrightness 0.1;
_light setLightAmbient[0.3, 0.3, 0.3];
_light setLightColor[1.0, 1.0, 1.0];
_light lightAttachObject [Licht, [1,1,1]]

```

Zweite Methode zur Erstellung einer Lichtquelle:

1. Man setze eine Spielfigur und gebe ihr einen markanten (langen) Namen.
2. Spiel im Editor speichern, mit Alt+Tab in den Missionsordner wechseln und die missions.sqm mit Editor öffnen.
3. Die Positionsdaten der Spielfigur (Bsp: position[]={2607.723,14.381,2832.153};) kopieren. Diese Positionsdaten in den createVehicleLocal Befehl eingeben:

(Bsp: `_light = "#lightpoint" createVehicleLocal [2607.723,14.381,2832.153];`)

4. Den zweiten Wert

(Bsp: `_light = "#lightpoint" createVehicleLocal [2607.723,14.381,2832.153];`) löschen und am Schluss den Höhenwert (zB 5) anfügen:

(Bsp: `_light = "#lightpoint" createVehicleLocal [2607.723,2832.153,5];`)

5. Resultat: Eine Lichtquelle wird auf der Position in 5m Höhe erstellt. Im Editor ist das setzen eines unsichtbaren H als Lichtposition nicht mehr notwendig.

Beispiel des light.sqs Scriptes nach der zweiten Methode:

```
_light = "#lightpoint" createVehicleLocal [2607.723,2832.153,5];
_light setLightBrightness 0.1;
_light setLightAmbient[0.3, 0.3, 0.3];
_light setLightColor[1.0, 1.0, 1.0];
```

Erläuterung der Befehle:

a) Erstellung der Lichtquelle

`_light = "#lightpoint" createVehicleLocal [0,0,0];`

b) Lichtquelle wird auf das Object I gesetzt.

Dabei gibt der erste Wert (wenn positiv) die Position der Lichtquelle rechts vom Object an.
(wenn negativ) die Position der Lichtquelle links vom Object an.

Der zweite Wert gibt (wenn positiv) die Entfernung der Lichtquelle an (hinter Object)
(wenn negativ) die Entfernung der Lichtquelle an (vor Object)

Der dritte Wert gibt (wenn positiv) die Höhe der Lichtquelle bezüglich des Objects an.
(wenn negativ) die Tiefe der Lichtquelle bezüglich des Objects an.

`_light lightAttachObject [Licht, [0, 0, 0]]`

c) Mit diesem Befehl kann man den Helligkeitswert der Lichtquelle steuern. Dabei ist der Wert 0.1 sehr dunkel, der Wert 10 wäre sehr hell.

`_light setLightBrightness 1;`

d) Hiermit wird der Farbanteil der Lichtquelle bestimmt. Der erste Wert gibt den Rotanteil an. Der zweite Wert gibt den Grünanteil an. Der dritte Wert gibt den Blauanteil an.

ACHTUNG! Die Farbe der Lichtquelle wird nicht mit diesem Befehl bestimmt. Dieser Befehl steuert nur den Farbcharakter der angeleuchteten Umgebung.

`_light setLightAmbient[0.0,1.0, 0.0];`

Die Umgebung würde also in diesem Beispiel Grün angestrahlt

e) Dieser Befehl bestimmt die Farbe der Lichtquelle. Der erste Wert bestimmt den Rot, der zweite den Grün und der dritte Wert den Blauanteil

`_light setLightColor[1.0, 0.0, 0.0];`

Die Lichtquelle wäre in diesem Beispiel Rot

Dies ist ein Scriptbeispiel indem sämtliche Parameter durch die Init Zeile bestimmt werden können:

Folgendes in die Init des Triggers oder der Init.sqs oder des Objektes eintragen:

`[Name,x,y,z,rot,gruen,blau,Lichtstärke,rotUmgebung,gruenUmgebung,blauUmgebung] exec "light.sqs"`

Bsp: `[Licht,0,0,3,1,1,1,0.3,1,1,1] exec "light.sqs"`

Folgendes in der light.sqs einfügen:

```

_objekt = _this select 0
_x = _this select 1
_y = _this select 2
_z = _this select 3
_rot = _this select 4
_gruen = _this select 5
_blau = _this select 6
_hell = _this select 7
_rotu = _this select 8
_gruenu = _this select 9
_blauu = _this select 10

_light = "#lightpoint" createVehicleLocal [0,0,0];
_light setLightBrightness _hell;
_light setLightAmbient[_rotu, _gruenu, _blauu];
_light setLightColor[_rot, _gruen, _blau];
_light lightAttachObject [_objekt, [_x,_y,_z]]
Exit

```

Fliegen an der Leiche

Eine nette Spielerei ist es auch wenn man an einer Leiche Fliegen erstellt. Sehr authentisch. Dazu erstellt eine Leiche, entweder unter Leer Objekte oder nimmt einen Kameraden dem ihr dann mit setdamage1 das Lebenslicht auspustet und schreibt in dessen Init Zeile [this] exec "leiche.sqs". Erstellt ein Skript im Missionsortner, Name leiche.sqs.

Quelltext:

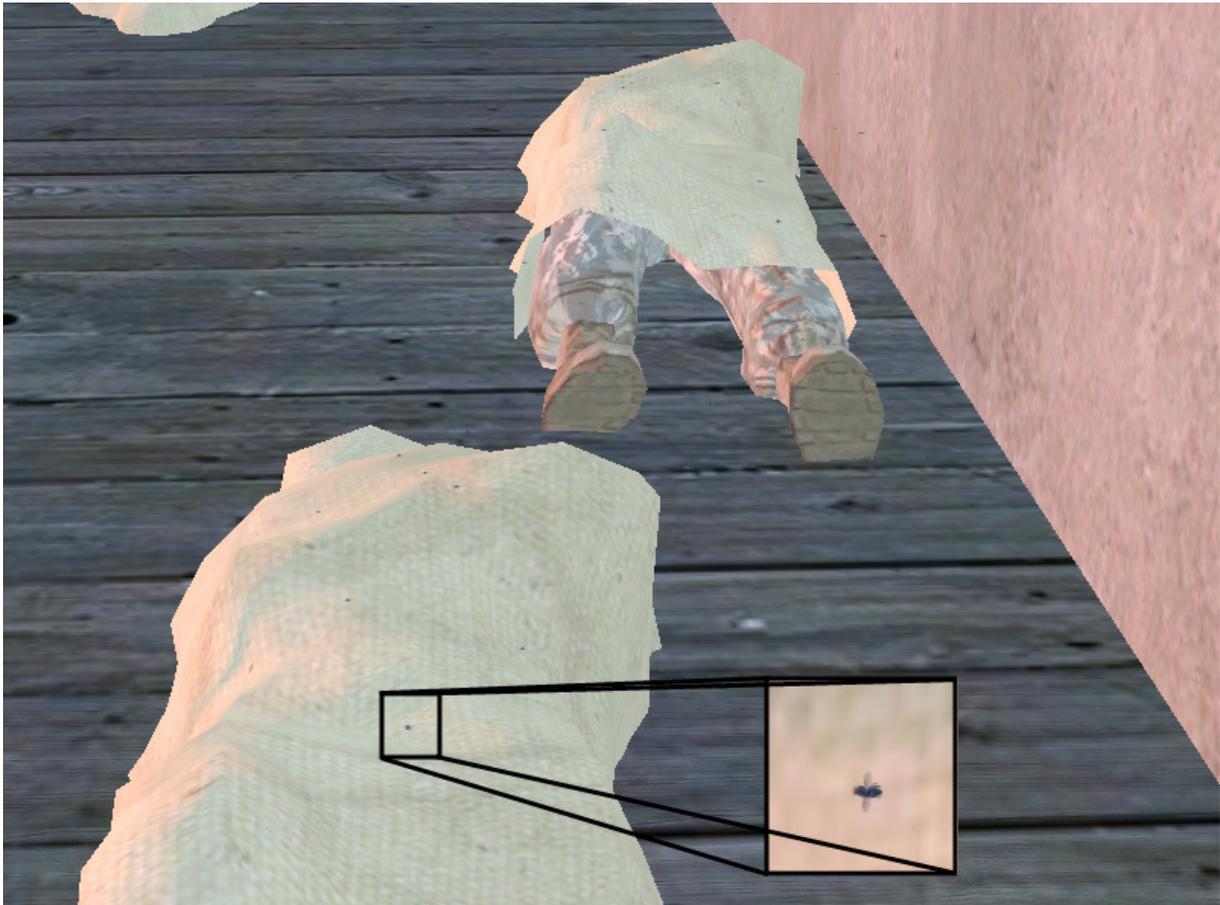
```

_unit = _this select 0;

_i = 0;
#loop
_x = (getPos _unit select 0) + 0.5 - random 1;
_y = (getPos _unit select 1) + 0.5 - random 1;
_z = (getPos _unit select 2) + 1.5 - random 1;
Name="HouseFly" camcreate [_x, _y, _z];
_i = _i + 1
? _i < (10 + random 3) : goto "loop"

```

Dieses Skript erstellt jetzt 10 bis 15 Fliegen an der Position der Leiche. Diese Bewegen sich automatisch auf und ab. Mann muss jedoch genau hinsehen.



Diese kleinen Freunde vom toten Fleisch fliegen dann über der Leiche, oder setzen sich auf den Körper. Eine lästige Angelegenheit.

Die Tierwelt

Jeder Spieler der in einem MP Spiel gestorben ist kennt die Möwe. Dieser kleine Vogel in den der Geist des Spielers respawned wird, kann auch extern erstellt werden. Die Möwe ist nichts anderes als ein Vehikel. Um eine Möwe zu erstellen wird in einem Auslöser oder in einer Initzeile etc . `Name="seagull" camcreate getPos player;` jetzt wird eine Möwe direkt an der Position der Player Einheit erstellt. Die Möwe sitzt jedoch am Boden und bewegt sich nicht. Dem kann Abhilfe geleistet werden in dem man der Einheit eine andere Höhe gibt.

`Name="seagull" camcreate [getPos player select 0, getPos player select 1, (getPos player select 2) + 10];`

Jetzt wird die Möwe über der Position des Spielers erstellt, jedoch in einer Höhe von 10 Metern. Die Möwen können auch eine Angabe `[X,Y,Z]` erstellt werden. Das macht sich in Videos besonders gut. Eine Möwe kann dann über Befehle durch die Gegend geschickt werden.

Weitere Tiere in ArMA sind:

	<p>Classname: SeaGull</p> <p>Artbestimmung: Regenpfeiferartige Charadriiformes Möwen Laridae Möwe Larus spec</p>
	<p>Classname: Hawk</p> <p>Artbestimmung: Greifvögel Falconiformes Habichtartige Accipitridae Habicht Accipiter gentilis</p> <p>(Artbestimmung nicht sicher)</p>
	<p>Classname: DragonFly</p> <p>Artbestimmung: Libellen Odonata Großlibellen Anisoptera Edellibellen Aeshnidae Blaugrüne Mosaikjungfer Aeshna cyanea</p> <p>(Artbestimmung nicht sicher)</p>

	<p>Classname: Butterfly</p> <p>Artbestimmung: Schmetterlinge Lepidoptera Edelfalter Nymphalidae Fleckenfalter Nymphalinae Kleiner Fuchs <i>Aglais urticae</i></p>
	<p>Classname: HouseFly</p> <p>Artbestimmung: Zweiflügler Diptera Fliegen Brachycera Echte Fliegen Muscidae Stubenfliege <i>Musca domestica</i></p>
	<p>Classname: HoneyBee</p> <p>Artbestimmung: Hautflügler Hymenoptera Tailenwespen Apocrita Bienen Apiformes Apidae - Honigbienen <i>Apis mellifera</i></p>

	<p>Classname: Mosquito</p> <p>Artbestimmung: Zweiflügler Diptera Mücken Nematocera Stechmücken Culicidae Stechmücke Culex pipiens</p> <p>Sieht aus wie ne HoneyBee!! BI! Von euch muß wol noch mal jemand in den Bio Unterricht!</p>
---	--

Des Weiteren gibt es noch einen Hasen und einen Wolf. Diese Tiere sind in ArMA vor gefertigt, es fehlen jedoch die Config und die animations Dateien.

	
Hase	Wolf

Zum erstellen der Tierarten einfach `Name="seagull" camcreate [X,Y,Z]`; gegen die gewünschte Art austauschen.

Dialoge

Es gibt die Möglichkeit eigene Dialoge einzubringen. Aber was ist das eigentlich. Hier ein Bild meines experimentellen Artillerie Dialog.



Hier sind 13 Buttons abgebildet welche in diesem Fall Skripte starten. Über ein solches Menue kann in diesem Fall der Einschlagspunkt geändert werden und der Abschuss der Artillerie Batterie koordiniert werden. Bei Allen Dialogen wird jedoch schnell klar, dass es sich hierbei um viele Einträge in die description.ext handelt. Dabei ist es Ratsam diese aus zu lagern und die externe Datei von Typ *.hpp mit `#include "*.hpp"` in die description.ext ein zu binden.

Beispiel einer description.ext:

```
class Header
{
    gameType = Coop;
    minPlayers = 2;
    maxPlayers = 30;
};
OnLoadintroTime = true
OnLoadMissionTime = true
Onloadintro = Alex presents
onLoadMission = Ambush

minScore=500;
avgScore=2500
maxScore=3000
```

```
ShowGPS = 1  
ShowCompass = 1  
ShowRadio = 1  
ShowMap = 1  
ShowNotePad = 1  
ShowWatch = 1  
ShowDebriefing = 1
```

```
respawn = 3  
respawnvehicle = 3  
respawnVehicleDelay = 10  
respawndelay = 6  
disabledAI = 1
```

```
#include "Ari.hpp"
```

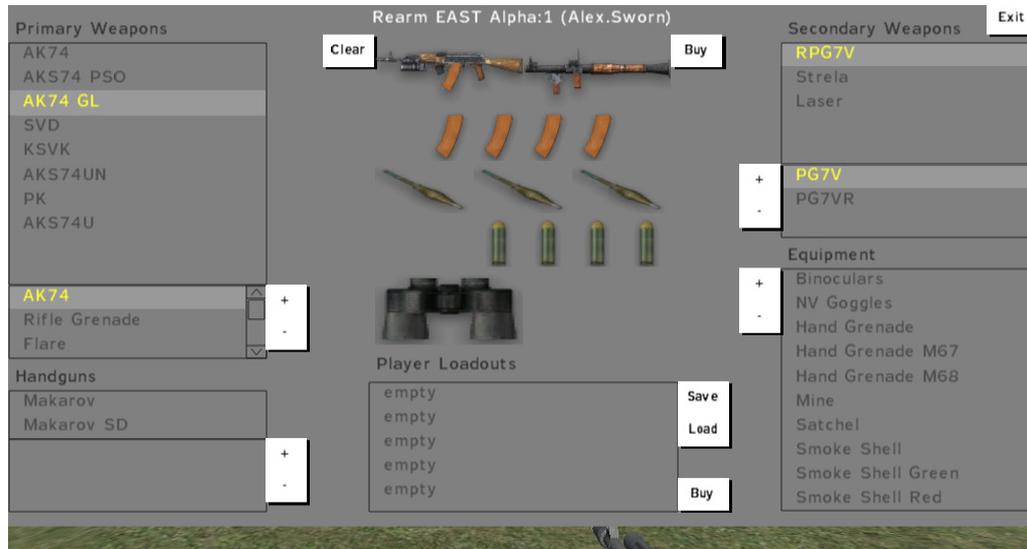
Natürlich lassen sich auch die anderen Einstellungen auslagern. Siehe Elegante Lösung.

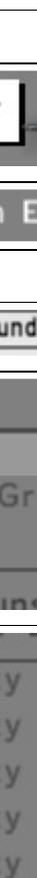
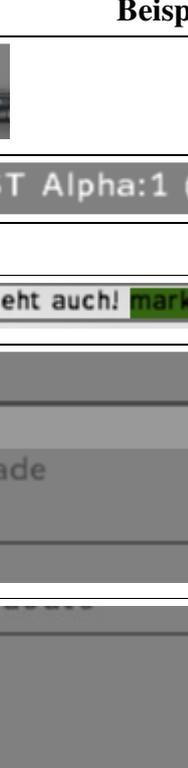
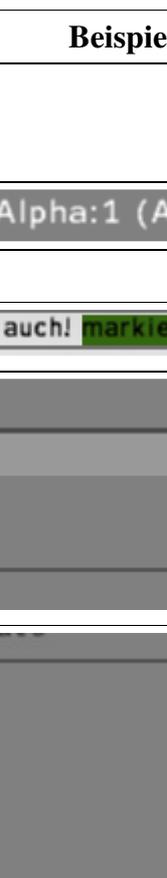
{Es wird gearbeitet. Es fehlen uns zur Zeit Informationen}



Button & co

Um Dialoge zu nutzen müssen diese zuerst in der description.ext definiert werden.



Bezeichnung, Typclass	Beispiel
Button	
Static Text	
Active Text	
Edit Box	
Drop Down Menues	
Listbox	

Statischer Text feste Texte ohne Funktion CT_STATIC

Um Dialoge zu nutzen müssen diese zuerst in der **description.ext** definiert werden.

Aktiver Text Bedienbarer Text CT_ACTIVETEXT

Der Unterschied zu „Statischer Text“ ist die Möglichkeit diesen mit der Maus anzuklicken. Dabei wird eine definierte Aktion ausgeführt.

Buttons Knöpfe CT_BUTTON

Ein Button kann sich im Gegensatz zu einem Aktiver Text bei der Aktivierung bewegen. Genau wie bei einem Aktiver Text wird eine definierte Aktion ausgeführt.

Textfelder Eingabefelder CT_EDIT

Textfelder sind Felder in denen der Spieler die Möglichkeit hat Texteeinzugeben.

ListBoxen CT_LISTBOX

Bei einer Listbox handelt es sich um eine Auflistung von Einträgen. Wenn diese eine bestimmte Anzahl überschreiten werden diese mit einer Scrollfunktion ausgestattet.

ComboBoxen CT_COMBO

Funktioniert ähnlich wie eine ListBox. Allerdings sind alle Einträge sofort sichtbar und der User braucht nicht zu scrollen.
mehr Infos in der nächsten Version.

{Es wird gearbeitet. Es fehlen uns zur Zeit Informationen}



Multiplayer

Respawn

Der größte Unterschied zwischen einer Einzelspielermission und einer Mehrspielermission ist die Möglichkeit, eine Spielerfigur mehrfach zu aktivieren. (Wieder an einer festgelegten Position auferstehen lassen). Es gibt verschiedene Formen dieses Auferstehens.

Diese werden in der description.ext vereinbart.

respawn= X ;	Hier wird die Art des Respawn vereinbart
respawnvehicle= X ;	Hier wird die Art des Respawn für Fahrzeuge vereinbart
respawnvehicleDelay= X ;	Hier wird die Zeit, die vor dem Respawn der Vehicle vergeht, vereinbart
respawnDelay= X ;	Hier wird die Zeit, die vor dem Respawn vergeht, vereinbart
disabledAI= X ;	Hier wird vereinbart ob die nicht besetzten Slots von der AI besetzt werden oder nicht

Es gibt die Möglichkeit, Einheiten in verschiedener Form zu erneuern. Mögliche Werte :

- | | |
|---|---|
| 0 | kein Respawn |
| 1 | Respawn als Möwe |
| 2 | Respawn am Todesplatz |
| 3 | Respawn an einem gesetzten Marker |
| 4 | Respawn in der Gruppe, wenn eine als spielbar gekennzeichnete Einheit vorhanden ist.
Dies geht nur wenn disabledAI= 0 ; ist und noch eine weitere Einheit frei ist. Ansonsten Respawn als Möwe. |
| 5 | Respawn in einer freien Einheit der selben Seite, wenn eine weitere Einheit als spielbar gekennzeichnet ist. Dies geht nur wenn disabledAI= 0 ; ist und noch eine weitere Einheit frei ist. Ansonsten Respawn als Möwe. |

Die Namen der Respawnmarker sind:

respawn_west	Westmarker
respawn_east	Ostmarker
respawn_guerrilla	Widerstandmarker
respawn_civilian	Civilmarker

Eine weitere Möglichkeit der Namensvergabe ist respawn_name der Einheit. Dieser Punkt ist dann der Respawnpunkt für den Anführer und die Soldaten seiner Gruppe. Damit wird es Möglich schnell verschiedene Punkte für die Gruppen zu bestimmen.

Es können auch weitere Marker verwendet werden.

respawn_west_1 ; respawn_west_2; ... Westmarker

Die jeweiligen Einheiten sollten jedoch dem Marker zugewiesen worden sein. Oder es wird ein Marker verwendet welcher auf der Insel seine Position verändert und somit den Punkt der Auferstehung bestimmt. Ein Auslöser startet ein Skript welches den Marker respawn_west an eine Position bringt.

Multiplayer und Server Installation

Man unterscheidet hierbei:

- **Dedizierter Server** (engl. dedicated server) ist ein Server, der nur für eine Aufgabe abgestellt wird

- **Host**: Ein Rechner stellt über eine Direktverbindung (LAN oder Internet) zu einem anderen Rechner den Kommunikationspartner dar. Dabei wird das Spiel auf dem Host Rechner gestartet und gesteuert

Für einen Host über ein LAN Netzwerk sollte der Rechner gute Spielvoraussetzungen haben da er nicht nur die Spieldarstellung des Spielers, sondern auch noch sämtliche Spieldaten für die Mitspieler erstellen und verwalten muss.

Um ein Host über Lan zu erstellen bedarf es nur ein paar Rechner die mit dem Host Rechner verbunden sind.

Einen Dedicat Server kann man auch im Lan wie auch im Internet betreiben . Die Installation ist die gleiche.

Um einen dedi Server über Internet zu erstellen benötigt man einen Rechner mit einer guten upload Verbindung, die aktuelle Version des Dedicaten Servers für Arma und natürlich das Originalspiel.

Die aktuelle Version des Dedicated Servers entpackt man in das Verzeichnis das wir als Beispiel

c:\Armaserver nennen.

Dann kopiert man das gesamtes Arma Assault Spiele Verzeichnis in das c:\Armaserver Verzeichnis.

Nun braucht man um den arma Server zu Starten eine Sogenannte Configdatei: server.cfg .

Die server.cfg sagt dem Server was er tun soll. Hierfür gibt es Befehle die ich nun erläutern werde. Als erstes muss man der Arma Server exe sagen welche Configdatei geladen werden sollen: Bsp:

C:\armaserver\Arma_Server.exe -name=llbrig -config= server.cfg -port 2303

-config=<config_file>	Sucht das richtige Configfile aus.
-port=<port_number>	Vergibt den Port welchen das Spiel benutzt Standart: Port 2302 Achtung checkt ob die Firewall den Port offen hat. Für Internet: ob euer Router diesen Port geöffnet hat. Wenn dieser nicht offen ist müsst ihr diesen in eurem Router einstellen.
-name=<Profil_Name>	Profile Name das der Server nutzen soll(Diese Befindet sich im Ordner C:\Dokumente und Einstellungen\dein_username\Eig ene Dateien\Arma Other Profiles**)

** Dieses Profile gibt an, ob auf dem Server Cadet oder Veteran Modus gespielt wird. Es muss gesondert angelegt werden in Arma Assault.!!!!

password = <Passwort>;	Passwort Schutz für den Server Standart: wenn dieser Befehl nicht in der
------------------------	---

	Config ist, ist kein Passwort vergeben.
passwordAdmin = <Admin_Passwort>;	Passwort wird gesetzt für Admin.(kann Mission wechsel und Player kicken. Standart: Befehl nicht in der Config keine Admin Rechte vergeben.
hostname = "<Servername>";	Server Name der sowohl im Gamespy Netz wie auch beim Einloggen im Server angezeigt wird.
motd[]= { "<1st MOTD Line>", "<2nd MOTD Line>", "<Last MOTD Line>" };	Message of the day (MOTD) Nachricht des Tages. Hier Kann man als Beispiel reinschreiben wer den Server Betreibt usw. Standart:Nicht vergeben keine MOTD Nachricht beim einloggen.
motdInterval=<interval_in_Sekunden>;	Interval in der die MOTD Linen angezeigt werden. Standart: 5 Sekunden wenn nicht vergeben in der Config.
voteThreshold=<threshold>;	Anzahl der Spieler in % ,die für einen erfolgreichen Vote gebraucht werden. Standart: wenn nicht vergeben ist die % Zahl 0.5(Mehr als die Hälfte muss voten.)
reportingIP="armedass.master.gamespy.com";	Befehl um den Gamespy net mitzuteilen das es den Server gibt.(Man macht den server damit öffentlich) Standart:wenn nicht vergeben in der Config wird der Server nicht öffentlich gemacht.
voteMissionPlayers=<Nummer>;	Die Menge der Spieler die connectet sein müssen bevor man eine Mission aussuchen kann. Standart: wenn Befehl nicht in Config 1Spieler
Kickduplicate=<1>	Arma Id Checker.Wenn Spieler mit der Gleichen Id connecten werden sie gekickt. Standart:wenn nicht vergeben können gleiche Ids nicht gekickt werden.

MaxMsgSend=<limit>;	Maximum Nummer der Chat Nachrichten die aufeinmal verschickt werden können.(bei zu hoher Nummer wird der Server laggie und es könnte zu Beeinträchtigungen des Spielverlaufskommen kommen, des weiteren müsst ihr hierbei auf eure Bandbreite achten) Standart: Wenn nicht vergeben in der Config sind es 128 Nachrichten
MaxCustomFileSize=<Grösse_in_Bytes>	User Custom Faces File Grösse. Ist es grösser als der Wert in Bytes wird der User Gekickt.

Beispiel Config

```

-
passwordAdmin = "*****"; // Admin Passwort!
hostname="LLBRIG31 DED.ARMA Server"; //ServerName!
MaxPlayers=100; // Setz Max. Nummer an Spielern!
MaxCustomFileSize=200000; // Max custom Face File Grösse!
motdInterval=5; // MOTD Linen Interval!
password = "*****"; // Passwort
motd[]=
{
    "Welcome to LUFTLANDEBRIGADE 31 DED.ARMA Server.",
    "Hosted by EVILMADCAT.Spiele Fair, hoere auf deinen Vorgesetzten ;-)",
}; // MOTD
reportingIP=" armedass.master.gamespy.com"; // Veröffentlicht den Server im Gamespy
Netz!
voteThreshold=0.1; // Nummer auf % votes die gebraucht
werden!
voteMissionPlayers=1; // Mindestanzahl ab der Vote aktiviert ist
kickduplicate=1; // Kick Doppelte ID's!
equalModRequired=1; // Abstimmung der gleichen
Versionsnummer

```

Desweiteren wenn ein admin Passwort vergeben ist können folgende Befehle genutzt werden:

#login <password>	Login Als Server Administrator(Admin)
#logout	Logout des Server Admin
#init	Reload Des Server Config File z.b: #init -config <Name>.
#kick <SpielerName>	Kickt den Player mit dem Namen
#kick <Spieler_number>	Kickt den Spieler mit der Id
#restart	Restart der Mission
#reassign	Geht zurück zum Missionsbriefing
#mission <mission_Name>	Sucht Mission mit diesen Namen aus
#shutdown	Fährt den Server runter
#userlist	Stellt eine Liste aller User mit Ids und Namen dar

XML

In der Squad.xml sind persönliche Daten eines Spielers oder einer Gruppe von Spielern festgelegt. In dieser Datei sind die Namen und die IDs der Spieler angegeben, sowie die Möglichkeit ein Logo im Spiel zu zeigen. Dieses Logo wird dann auf Einheiten angezeigt. So können sich Clanspieler schneller und einfacher erkennen. Außerdem sieht es einfach gut aus. Weitergehend ist noch zu erwähnen, dass die Datei auf einem FTP Server liegen muss und frei erreichbar sein sollte. Ansonsten läuft diese nicht. Der Link zu dieser Datei wird dann unter URL im Benutzerkonto eingetragen.
Quellcode Squad.xml für einen User:

```
<squad nick="LLBrig 31">
<name>Luftlandebrigade 31</name>
<email>llbrig31@googlemail.com</email>
<web>http://www.llbrig31.de.tp</web>
<picture>logo.paa</picture>
<title>LLBrig 31</title>
  <member id="12345612" nick="x-reh">
    <name>Tim</name>
    <email>haltsmaul@googlemail.com</email>
    <icq>123-123-123</icq>
    <remark>Des Teufels Finanzberater</remark>
  </member>
</squad>
```

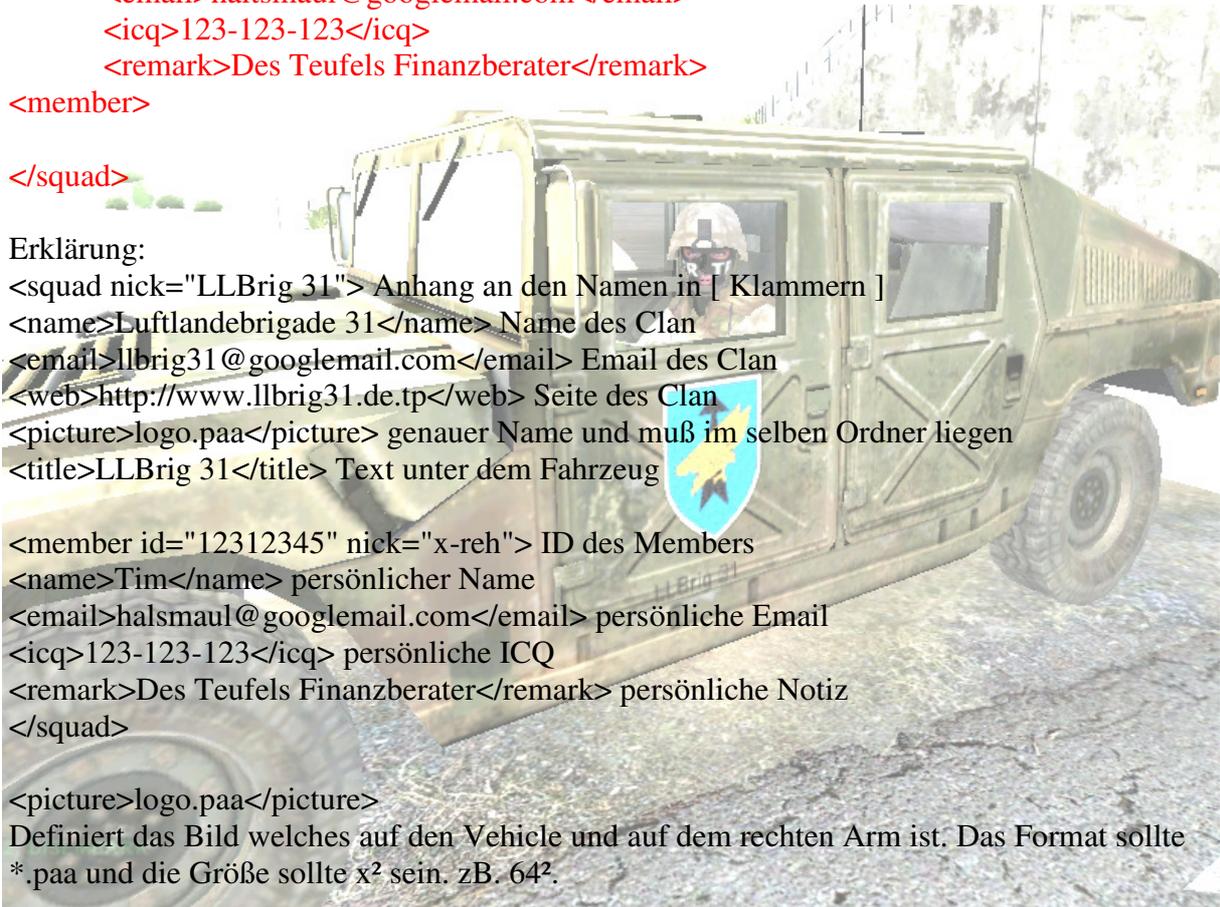


Erklärung:

```
<squad nick="LLBrig 31"> Anhang an den Namen in [ Klammern ]
<name>Luftlandebrigade 31</name> Name des Clan
<email>llbrig31@googlemail.com</email> Email des Clan
<web>http://www.llbrig31.de.tp</web> Seite des Clan
<picture>logo.paa</picture> genauer Name und muß im selben Ordner liegen
<title>LLBrig 31</title> Text unter dem Fahrzeug
  <member id="12312345" nick="x-reh"> ID des Members
  <name>Tim</name> persönlicher Name
  <email>haltsmaul@googlemail.com</email> persönliche Email
  <icq>123-123-123</icq> persönliche ICQ
  <remark>Des Teufels Finanzberater</remark> persönliche Notiz
</squad>
```

```
<picture>logo.paa</picture>
```

Definiert das Bild welches auf den Vehicle und auf dem rechten Arm ist. Das Format sollte *.paa und die Größe sollte x² sein. zB. 64².

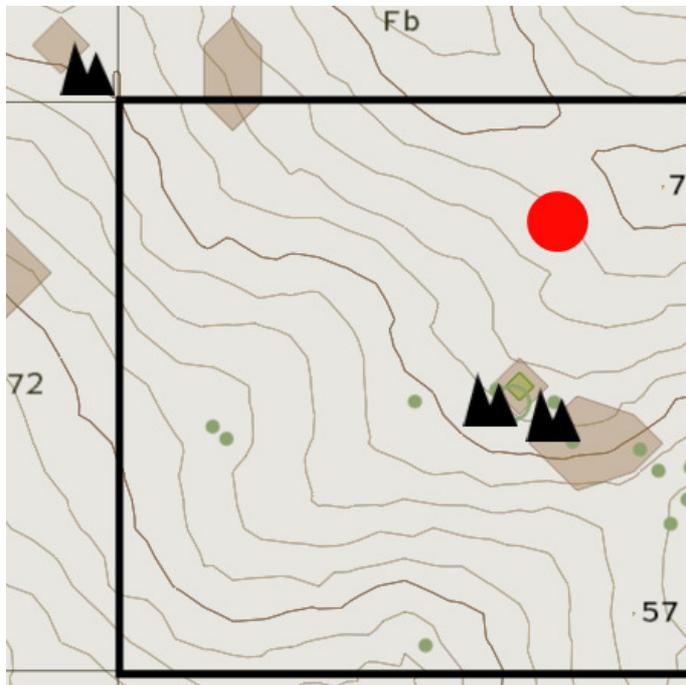


Netzwerkoptionen

- port= Wählt den Port aus, mit dem der dedizierte Server gehostet werden soll.
- password= Passwort um auf dem Server zu joinen.
- ranking= Unbekannt
- config= Ermöglicht eine server.cfg zu spezifizieren für serverspezifische Einstellungen wie z.B. Adminpasswort und Missionauswahl.
- host Startet einen nicht dedizierten Server (Im Browser an dem „auf“ zu erkennen. Beispiel: „ArmA_Server_1.02 auf PC_Klaus“ zu erkennen).
- server Startet einen dedizierten Server (wird bei einer speziellen .exe für einen dedizierten Server nicht benötigt.)
- connect= Client seitig, hinter dem = kommt die gewünschte IP des Servers, auf dem man connecten möchte.
- name= Client seitig, eigener Username
- profiles= Alternative Platzierung für userabhängigen Content (bessere Übersetzung? „Alternative placement for per-user content.“)
- netlog Aktiviert das speichern des ArMA MP Datenverkehr (Traffic) in einer LogDatei

Die Koordinaten

Viele MP Spiele gestalten sich als kompliziert, da sich Leute verlaufen und Treffpunkte nicht finden. Dann ließt man im Chat oder hört im TS öfter die Worte wo seid ihr. Das Kartenraster bei ArMA ist zu groß geraten. Die Koordinate FB 72 ist als Treffpunkt recht grob. Ein Treffen bei Nacht und Nebel daher nur schwer zu machen. Die Lösung ist das Unterteilen dieses Quadranten in 100 kleine Felder. Dann wird aus der XY Koordinate FB3 727. Diese Angabe würde dann FB mit dem 3 Feld und 72 das 7 Feld. Eine Positionsangabe dieser Art ist erheblich einfacher zu finden.



NATO-Alphabet

Um Informationen im Funkverkehr besser zu übermitteln werden einzelne Buchstaben als Wörter weiter übergeben. Dadurch werden Fehler verhindert. Macht in MP Spielen besonders

A - ALFA
B - BRAVO
C - CHARLIE
D - DELTA
E - ECHO
F - FOXTROT
G - GOLF
H - HOTEL
I - INDIA
J - JULIETT
K - KILO
L - LIMA
M - MIKE

N - NOVEMBER
O - OSCAR
P - PAPA
Q - QUEBEC
R - ROMEO
S - SIERRA
T - TANGO
U - UNIFORM
V - VICTOR
W - WHISKEY
X - XRAY
Y - YANKEE
Z - ZULU

Insel erstellen

Der Traum jedes Missionsdesigners ist es nicht nur einzelne Einheiten oder Objekte zu platzieren oder zu bewegen, sondern ganze Inseln aus dem Meer erscheinen zu lassen. Das Gelände zu formen, es anzupassen. Jeder der schon einmal versucht hat eine MG Stellung perfekt zu platzieren wird sicherlich verstehen was es bedeutet das Land dem MG anzupassen. Was unsere realen Kollegen mit Schaufel und Spaten machen, geht in der ArMA Welt per Mausclick. Schnell, präzise und ohne große Anstrengungen. Was aber genau ist eine Insel? Ein Addon im dem Sinne eines Objektes ist sie wohl nicht. Sie besteht aus einer Gelände Struktur etwas Farbe und einigen Objekten.

Die Insel PBO

Die Insel PBO ist nichts anderes als die gepackten Daten der Insel. Dies geschieht mit einem externen Tool. makepbo oder pboextrakt sind da einige. Vergleichbar mit der Mission.pbo.

Der Inselordner

Die Insel besteht aus mindestens zwei Teilen. Die *.wrp Datei und einer config.cpp. Es können weitere Dateien wie Skripte für Funktionen auf der Insel eingebunden werden. Sowie Dateien für Platzierungen von Objekten, Texturen oder ganze Gebäude *.p3d können eingebunden werden. Es gibt da keine Grenzen.

Config.cpp

In der config.cpp sind der Name der Insel, die Dateiverwaltung, evtl Texturen, Intros, Skripte und die Namen auf der Karte aufgeführt. Dies wird durch Vereinbarungen in der jeweiligen class **** durch geführt. Alle Einstellungen zu Start Wetter, Tiere, Pflanzen, sowie alle weitem Optionen sind so bestimmt.. (Wichtig: Class namen dürfen nur einmal verwendet werden!)

Class CfgPatches. Hier werden Namen der Insel, benötigte Version von ArmA sowie benötigte Addons vereinbart.

//class Insel, Version, Name, Addons.

```
class CfgPatches {
    class Pacificav1 {
        units[] = {"Pacifica"};
        weapons[] = {};
        // benötigtes ArmA z.B. v 1.4 etc.
        requiredVersion = 0.100000;
        // benötigte Addons. Werden beim starten von ArmA abgefragt.
        requiredAddons[] = {"CAData", "CABuildings", "CARoads", "CARocks"};
    };
};
```

Hier sind die Start Parameter für Zeit Wetter etc. vereinbart. startTime ist die Zeit beim öffnen der Insel im Editor, startDate ist das Datum, startWeather ist das Wetter beim Starten, startFog ist der Nebelwert, forecastWeather ist das kommende Wetter, forecastFog ist der kommende Nebel.

//Insel Time Daten

```
startTime = "12:05";
startDate = "22/05/2007";
startWeather = 0.100000;
startFog = 0.000000;
forecastWeather = 0.100000;
forecastFog = 0.000000;
```

Class namen {};

Einer Karte dürfen Namen für Orte auf der Insel nichtfehlen. Oder es sollen Symbole abgebildet werden. Das kann in class Names vereinbart werden. Name ist der eingetragene Name. position die Position auf der Karte. (X,Y,Z)

```
/**
//      Eingetragene Namen auf der Karte (X, Y, Z) keine ä ü ö etc
// ****
class Names {
    class Flug1 {
        name="Flughafen";
        position[]={6320.593262, 3862.714355, 20};
    };
    class Ruien3 {
        name="Ruinenstadt Jabares";
        position[]={5832.658691, 6875.824219, 55};
    };
};
```

Name + Symbol auf der Karte. type ist der Typ des Markers. radius A und radius B sind die Größen des Markers.

```

//*****
//      Eingetragene Namen, Symbole auf der Karte (X, Y, Z) keine ä ü ö etc
// *****
class Names {
    class Sara_Rashidah {
        name = "Rashidah";
        position = {9585.790039, 11109.200195};
        type = "NameVillage";
        radiusA = 100;
        radiusB = 100;
    };

    class Sara_Balmopan {
        name = "Balmopan";
        position = {8035.759766, 9456.250000};
        type = "NameVillage";
        radiusA = 100;
        radiusB = 100;
    };
};

```

Quellcode einer config.cpp einer Insel.

```

// *****
// *** Dies ist meine Test Insel Pacifica (OFP) ***
// ***      unofficial Addon          ***
// ***      Kartell V1.0              ***
// *****

//define Einstellungen
#define ReadAndWrite          0
#define ReadAndCreate        1
#define ReadOnly              2
#define ReadOnlyVerified     3

//class Insel, Version, Name, Addons.
class CfgPatches {
    class Pacificav1 {
        units[] = {"Pacifica"};
        weapons[] = {};
        // benötigtes ArmA z.B. v 1.4 etc.
        requiredVersion = 0.100000;
        // benötigte Addons. Werden beim starten von ArmA abgefragt.
        requiredAddons[] = {"CAData", "CABuildings", "CAMisc", "CAPlants", "CARoads", "CARocks"};
    };
};

//extern
class DefaultLighting;
class DefaultLighting_CA : DefaultLighting {};
class CfgWorlds {
    class DefaultClutter {
        scaleMin = 0.900000;
        scaleMax = 1.200000;
    };
};

```

```

//extern
class DefaultWorld;
class CAWorld : DefaultWorld {
    class Grid {};
};

class Pacificainself : CAWorld {
    access = 3;
    cutscenes[] = {"Pacifica_v1"};
    description = "Pacifica v 1.0";
    // kleines Bild welches der Editor in der Kartenwahl zeigt. Ist in 1.7 beta noch nicht da.
    icon = "";
    // Angabe des Ordners in dem die Daten der Map zu finden sind. Name\Name.wrp
    worldName = "\Pacifica\Pacifica.wrp";
    pictureMap = "";
    pictureShot = "";
    plateFormat = "ML$ - #####";
    plateLetters = "ABCDEFGHIJKLMNPRSTVXZ";
    latitude = -40.020000;
    longitude = 15;
    landGrid = 40;

    class Grid : Grid {
        offsetX = -4880;
        offsetY = -7480;

        class Zoom1 {
            zoomMax = 0.100000;
            format = "XY";
            formatX = "Aa";
            formatY = "00";
            stepX = 200;
            stepY = 200;
        };

        class Zoom2 {
            zoomMax = 10000000150474662000000000000000.000000;
            format = "XY";
            formatX = "A";
            formatY = "0";
            stepX = 2000;
            stepY = 2000;
        };
    };
};

//Insel Time Daten und ILS System
startTime = "12:05";
startDate = "22/05/2007";
startWeather = 0.100000;
startFog = 0.000000;
forecastWeather = 0.100000;
forecastFog = 0.000000;
seagullPos[] = {8897,4349,100};
ilsPosition[] = {0,0,0};
ilsDirection[] = {0,0.08,1};
ilsTaxiIn[]={};
ilsTaxiOff[]={};
centerPosition[]={9735,3964,0};

class ReplaceObjects {
};

class Sounds {
    sounds[] = {};
};

class Animation {
    vehicles[] = {};
};

```

```

};

class Lighting : DefaultLighting {};
clutterGrid = 1.110000;
clutterDist = 55;
noDetailDist = 40;
fullDetailDist = 5;
minTreesInForestSquare = 3;
minRocksInRockSquare = 2;

//Hier werden Pflanzen Steine etc vereinbart.
class clutter {

    class GrassGeneral : DefaultClutter {
        model = "ca\plants\clutter_grass_general.p3d";
        affectedByWind = 0.3;
        swLighting = 1;
            scaleMin = 0.75;
            scaleMax = 1.0;
    };

    class GrassFlowers : GrassGeneral {
        model = "ca\plants\clutter_grass_flowers.p3d";
    };

    class GrassLong : GrassGeneral {
        model = "ca\plants\clutter_grass_long.p3d";
        affectedByWind = 0.6;
        scaleMin = 0.6;
        scaleMax = 1.1;
    };

    class GrassSevenbeauty : GrassGeneral {
        model = "ca\plants\clutter_grass_sevenbaeuty.p3d";
        affectedByWind = 0.1;
        scaleMin = 0.7;
        scaleMax = 1.1;
    };

    class GrassYellow : GrassGeneral {
        model = "ca\plants\clutter_grass_yellow.p3d";
        affectedByWind = 0.1;
        scaleMin = 0.7;
        scaleMax = 1.1;
    };

    class GrassDesert : GrassGeneral {
        model = "ca\plants\clutter_grass_desert.p3d";
    };

    class ForestFern : GrassGeneral {
        model = "ca\plants\clutter_forest_fern.p3d";
        affectedByWind = 0.1;
        scaleMin = 0.9;
        scaleMax = 1.1;
    };

    class SmallRocks : GrassGeneral {
        model = "ca\rocks\clutter_stone_small.p3d";
        affectedByWind = 0;
        scaleMin = 0.9;
        scaleMax = 1.1;
    };

    class FlowersColor : GrassGeneral {
        model = "ca\plants\clutter_smetanka.p3d";
    };

    class FlowersWhite : GrassGeneral {

```

```

        model = "ca\plants\clutter_white_flower.p3d";
    };

    class MushroomsHorcak : GrassGeneral {
        model = "ca\plants\clutter_horcak.p3d";
        affectedByWind = 0;
        scaleMin = 0.85;
        scaleMax = 1.25;
    };

    class MushroomsPrasivka : MushroomsHorcak {
        model = "ca\plants\clutter_prasivky.p3d";
    };

    class MushroomsBabka : MushroomsHorcak {
        model = "ca\plants\clutter_babka.p3d";
    };

    class MushroomsMuchomurka : MushroomsHorcak {
        model = "ca\plants\clutter_muchomurka.p3d";
    };
};

class Subdivision {
    minY = 0.0;
    minSlope = 0.02;

    class Fractal {
        roughness = 5;
        maxRoad = 0.020000;
        maxTrack = 0.500000;
        maxSlopeFactor = 0.050000;
    };
    class WhiteNoise {
        roughness=5;
        maxRoad=0.1;
        maxTrack=0.5;
        maxSlopeFactor=0.025;
    };
};

class Ambient {
    class BigBirds {
        radius = 300;
        cost = "(1 + forest + trees) - ((2 * rain)) - houses) * (1 - night) * (1 - sea)";
        class Species {
            class Hawk {
                probability = 0.2;
                cost = 1;
            };
        };
    };

};

class Birds {
    radius = 170;
    cost = "(1 - night) * ((1 + (8 * sea)) - (6 * rain))";
    class Species {
        class Seagull {
            probability = 0.2;
            cost = 1;
        };
    };
};

class NoWindClutter {
    radius = 15;
    cost = "(20 * (windy factor [0.1, 0.2])) * meadow * (1 - rain) * (1 - sea) * (1 - forest) * (1 -
houses)";

    class Species {
        class FxWindPollen1 {
            probability = 1;
            cost = 1;
        };
    };
};

```

```

    };
};

class SmallInsects {
    radius = 3;
    cost = "(12 - 8 * hills) * (1 - night) * (1 - rain) * (1 - sea) * (1 - windy)";
};

class Species {
    class HouseFly {
        probability = "deadBody + (1 - deadBody) * (0.5 - forest * 0.1 - meadow * 0.2)";
        cost = 1;
    };
    class Mosquito {
        probability = "(1 - deadBody) * (0.2 * forest)";
        cost = 1;
    };
};

radius = 3;
class NightInsects {
    cost = "(9 - 8 * hills) * night * (1 - rain) * (1 - sea) * (1 - windy)";
    class Species {
        class Mosquito {
            probability = 1;
            cost = 1;
        };
    };
};

// *****
//   Eingetragene Namen auf der Karte (X, Y, Z) keine ä ü ö etc
// *****
class Names {
    class Flug1 {
        name="Flughafen";
        position[]={6320.593262, 3862.714355, 20};
    };
    class Ruinen3 {
        name="Ruinenstadt Jabares";
        position[]={5832.658691, 6875.824219, 55};
    };
};

class CfgWorldList {
    class Pacificainsel {};
};
// Bestimmen des Ablageordners des Intros.
class CfgMissions {
    class Cutscenes {
        class PacificainselIntro1 {
            directory = "...Pacifica\intro.Pacificainsel";
        };
    };
};
};

```

Das WRP Format

Im *.wrp sind alle Gelände Daten (Höhen) gespeichert, sowie alle verwendeten Objekte aufgeführt (Dessen IDs) sowie Daten zu Texturen der Insel. Dieses kann in Tools wie visitor3 oder Wrptool 0.095 bearbeitet werden. Der Umgang mit diesen Tools wird extra erklärt.

DatenBank

Westwaffen und Ausrüstung

Weapon Classname	Magazine Classnames	Ammo Classnames	Hit Damage	Bild
Sturmgewehre				
M16A2	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD	B_556x45_Ball B_556x45_Ball B_556x45_SD	8 8 7	
M16A4	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD	B_556x45_Ball B_556x45_Ball B_556x45_SD	8 8 7	
M16A4_ACG	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD	B_556x45_Ball B_556x45_Ball B_556x45_SD	8 8 7	
M4	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD	B_556x45_Ball B_556x45_Ball B_556x45_SD	8 8 7	
M4AIM	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD	B_556x45_Ball B_556x45_Ball B_556x45_SD	8 8 7	
M4A1	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD	B_556x45_Ball B_556x45_Ball B_556x45_SD	8 8 7	
G36K	30Rnd_556x45_G36	B_556x45_Ball	8	
G36C	30Rnd_556x45_G36	B_556x45_Ball	8	
G36A	30Rnd_556x45_G36	B_556x45_Ball	8	
Sturmgewehre SD				
M4A1SD	30Rnd_556x45_StanagSD 30Rnd_556x45_Stanag 20Rnd_556x45_Stanag	B_556x45_SD B_556x45_Ball B_556x45_Ball	7 8 8	
Sturmgewehre + Granaten				
M16A4_ACG_GL	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD FlareWhite_M203 FlareGreen_M203 FlareRed_M203 FlareYellow_M203 1Rnd_HE_M203	B_556x45_Ball B_556x45_Ball B_556x45_SD F_40mm_White F_40mm_Green F_40mm_Red F_40mm_Yellow G_40mm_HE	8 8 7 - - - - 12	
M4A1GL	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD	B_556x45_Ball B_556x45_Ball B_556x45_SD	8 8 7	

	FlareWhite_M203 FlareGreen_M203 FlareRed_M203 FlareYellow_M203 1Rnd_HE_M203	F_40mm_White F_40mm_Green F_40mm_Red F_40mm_Yellow G_40mm_HE	- - - - 12	
M16A4GL	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD FlareWhite_M203 FlareGreen_M203 FlareRed_M203 FlareYellow_M203 1Rnd_HE_M203	B_556x45_Ball B_556x45_Ball B_556x45_SD F_40mm_White F_40mm_Green F_40mm_Red F_40mm_Yellow G_40mm_HE	8 8 7 - - - - 12	
M4GL	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD FlareWhite_M203 FlareGreen_M203 FlareRed_M203 FlareYellow_M203 1Rnd_HE_M203	B_556x45_Ball B_556x45_Ball B_556x45_SD F_40mm_White F_40mm_Green F_40mm_Red F_40mm_Yellow G_40mm_HE	8 8 7 - - - - 12	
M16A2GL	30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD FlareWhite_M203 FlareGreen_M203 FlareRed_M203 FlareYellow_M203 1Rnd_HE_M203	B_556x45_Ball B_556x45_Ball B_556x45_SD F_40mm_White F_40mm_Green F_40mm_Red F_40mm_Yellow G_40mm_HE	8 8 7 - - - - 12	
Maschinen- Pistolen				
MP5A5	30Rnd_9x19_MP5 30Rnd_9x19_MP5SD	B_9x19_Ball B_9x19_SD	8 8	
MP5SD	30Rnd_9x19_MP5SD 30Rnd_9x19_MP5	B_9x19_SD B_9x19_Ball	8 8	
Maschinen- gewehre				
M249	200Rnd_556x45_M249 30Rnd_556x45_Stanag 20Rnd_556x45_Stanag 30Rnd_556x45_StanagSD	B_556x45_Ball B_556x45_Ball B_556x45_Ball B_556x45_SD	8 8 8 7	
M240	100Rnd_762x51_M240	B_762x51_Ball	9	
Scharfschützen- Gewehre				
M4SPR	20Rnd_556x45_Stanag 30Rnd_556x45_Stanag 30Rnd_556x45_StanagSD	B_556x45_Ball B_556x45_Ball B_556x45_SD	8 8 7	
M24	5Rnd_762x51_M24	B_762x51_Ball	9	

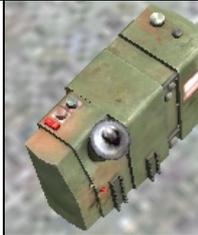
M107	10Rnd_127x99_m107	B_127x99_Ball_no Tracer	13	
Pistolen				
M9	15Rnd_9x19_M9 15Rnd_9x19_M9SD	B_9x19_Ball B_9x19_SD	8 8	
M9SD	15Rnd_9x19_M9 15Rnd_9x19_M9SD	B_9x19_Ball B_9x19_SD	8 8	
Raketen				
M136	M136	R_M136_AT	500	
JAVELIN	JAVELIN	M_Javelin_AT	800	
STINGER	STINGER	M_Stinger_AA	50	
Equipment				
Weapon Classname	Magazine Classnames	Ammo Classnames	Hit Damage	
Put?	PipeBomb TimeBomb Mine MineE	PipeBomb TimeBomb Mine MineE	1200 1200 1200 1200	
Throw?	SmokeShellRed SmokeShellGreen SmokeShell HandGrenadeTimed HandGrenade	SmokeShellRed SmokeShellGreen SmokeShell GrenadeHandTimed GrenadeHand	- - - 20 20	
Laserdesignator	Laserbatteries	-	-	
NVGoggles	-	-	-	
Binocular	-	-	-	
Vehicle- Weapons				
Weapon Classname	Magazine Classnames	Ammo Classnames	Hit Damage	Vehicle Classnames

UH60				
M134	2000Rnd_762x51_M134	B_762x51_Ball	9	UH60MG
M134_2	2000Rnd_762x51_M134	B_762x51_Ball	9	UH60MG
FFARLauncher	38Rnd_FFAR	R_Hydra_HE	30	UH60
AH6				
TwinM134	4000Rnd_762x51_M134	B_762x51_Ball	9	AH6 AH6_RACS
FFARLauncher	14Rnd_FFAR	R_Hydra_HE	30	AH6 AH6_RACS
AH1Z				
M197	750Rnd_M197_AH1	B_20mm_AP	20	AH1W
FFARLauncher	38Rnd_FFAR	R_Hydra_HE	30	AH1W
HellfireLauncher	8Rnd_Hellfire	M_Hellfire_AT	1000	AH1W
AV8B				
GAU12	300Rnd_25mm_GAU12	B_25mm_HE	15	AV8B AV8B2
BombLauncher	6Rnd_GBU12_AV8B	Bo_GBU12_LGB	5000	AV8B
SidewinderLaucher	4Rnd_Sidewinder_AV8B	M_Sidewinder_AA	70	AV8B2
A10				
MaverickLauncher	5Rnd_Maverick_A10	M_Maverick_AT	800	A10
GAU8	1350Rnd_30mmAP_A10	B_30mmA10_AP	40	A10
Tank				
M240_veh	1200Rnd_762x51_M240	B_762x51_Ball	9	M1Abrams
M256	20Rnd_120mmSABOT_M1A2 20Rnd_120mmHE_M1A2	Sh_120_SABOT Sh_120_HE	1000 80	M1Abrams

M2	100Rnd_127x99_M2	B_127x99_Ball	13	HMMWV50 Truck5tMG Stryker_ICV_ M2 LandroverMG RHIB RHIB2Turret M1Abrams M113 M113_RACS
TOWLauncher	2Rnd_TOW	M_TOW_AT	700	Stryker_TOW
TOWLauncherSingle	6Rnd_TOW_HMMWV	M_TOW_AT	700	HMMWVTOW
MK19	48Rnd_40mm_MK19	G_40mm_HE	12	HMMWVMK Stryker_ICV_ MK19 RHIB2Turret
M168	2100Rnd_20mm_M168	B_20mm_AA	40	Vulcan Vulcan_RACS
Artillerie				
M119	30Rnd_105mmHE_M119	a	50	M119

Ost Waffen und Ausrüstung

Weapon Classname	Magazine Classnames	Ammo Classnames	Hit Damage	Bild
Sturmgewehre				
AK47	30Rnd_545x39_AK	B_545x39_Ball	8	
AKS74U	30Rnd_545x39_AK	B_545x39_Ball	8	
AKS74PSO	30Rnd_545x39_AK	B_545x39_Ball	8	
Sturmgewehre SD				
AKS74UN	30Rnd_545x39_AKSD 30Rnd_545x39_AK	B_545x39_SD B_545x39_Ball	8 8	
Sturmgewehre + Granaten				
AK74GL	30Rnd_545x39_AK FlareWhite_GP25 FlareGreen_GP25 FlareRed_GP25 FlareYellow_GP25 1Rnd_HE_GP25	B_545x39_Ball F_40mm_White F_40mm_Green F_40mm_Red F_40mm_Yellow G_30mm_HE	8 - - - - 13	
Maschinen- Gewehre				
PK	100Rnd_762x54_PK	B_762x54_Ball	9	
Scharfschützen- Gewehre				

SVD	10Rnd_762x54_SVD	B_762x54_noTrace	9	
KSVK	5Rnd_127x108_KSVK	B_127x108_Ball		
Pistolen				
Makarov	8Rnd_9x18_Makarov 8Rnd_9x18_MakarovSD	B_9x18_Ball B_9x18_SD	8 8	
MakarovSD	8Rnd_9x18_Makarov 8Rnd_9x18_MakarovSD	B_9x18_Ball B_9x18_SD	8 8	
Raketen				
RPG7V	PG7V PG7VR	R_PG7V_AT R_PG7VR_AT	600 600	
STRELA	STRELA	M_Strela_AA	50	
Equipment				
Weapon Classname	Magazine Classnames	Ammo Classnames	Hit Damage	
Put?	PipeBomb TimeBomb Mine MineE	PipeBomb TimeBomb Mine MineE	1200 1200 1200 1200	
Throw?	SmokeShellRed SmokeShellGreen SmokeShell HandGrenadeTimed HandGrenade	SmokeShellRed SmokeShellGreen SmokeShell GrenadeHandTimed GrenadeHand	- - - 20 20	
Laserdesignator	Laserbatteries	-	-	
NVGoggles	-	-	-	
Binocular	-	-	-	
Vehicle- Weapons				
Weapon Classname	Magazine Classnames	Ammo Classnames	Hit Damage	Vehicle Classnames
PKT	1500Rnd_762x54_PKT	B_762x54_Ball	9	BRDM2
PKT	2000Rnd_762x54_PKT	B_762x54_Ball	9	BMP2 T72

				Mi17_MG
DSHKM	50Rnd_127x107_DSHKM	B_127x107_Ball	13	UAZMG T72
AGS30	29Rnd_30mm_AGS30	G_30mm_HE	13	UAZ_AGS30
KPVT	500Rnd_145x115_KPVT	B_145x115_AP	18	BRDM2
AZP85	2000Rnd_23mm_AZP85	B_23mm_AA	40	ZSU
2A42	230Rnd_30mmAP_2A42	B_30mm_AP	25	KA50
	230Rnd_30mmHE_2A42	B_30mm_HE	18	"
	250Rnd_30mmAP_2A42	B_30mm_AP	25	BMP2
	250Rnd_30mmHE_2A42	B_30mm_HE	18	"
D81	23Rnd_125mmSABOT_T7	Sh_125_SABOT	980	T72
	2	Sh_125_HE	80	"
	22Rnd_125mmHE_T72			
57mmLauncher	96Rnd_57mm	R_57mm_HE	30	Mi17
80mmLauncher	40Rnd_80mm	R_80mm_HE	50	KA50
R73Launcher	4Rnd_R73	M_R73_AA	-	Su34
S8Launcher	80Rnd_S8T	R_S8T_AT	-	
GSh301	180Rnd_30mm_GSh301	B_30mm_HE	18	
Ch29Launcher	4Rnd_Ch29	M_Ch29_AT	-	Su34B
S8Launcher	40Rnd_S8T	R_S8T_AT	-	
GSh301	180Rnd_30mm_GSh301	B_30mm_HE	18	
AT5Launcher	5Rnd_AT5_BRDM2	M_TOW_AT	700	BRDM2_ATG
	8Rnd_AT5_BMP2	M_AT5_AT	700	M BMP2
VikhrLauncher	12Rnd_Vikhr_KA50	M_Vikhr_AT	1000	KA50
D30	30Rnd_122mmHE_D30	Sh_122_HE	60	D30
SEARCHLIGHT	-	-	-	SEARCHLIG HT

Ausrüstung Fahrzeuge

Weapon Classname	Magazine Classnames	Vehicle Classnames
CarHorn	N/A	HMMWV UAZ Landrover_Closed Landrover Bus_city tractor hilux1_civil_1_open hilux1_civil_2_covered hilux1_civil_3_open
BikeHorn	N/A	N/A
TruckHorn	N/A	Truck5t Truck5tOpen Truck5tRepair Truck5tReammo Truck5tRefuel Ural UralOpen UralCivil

		UralCivil2
SportCarHorn	N/A	Skoda SkodaBlue SkodaRed SkodaGreen datsun1_civil_1_open datsun1_civil_2_covered datsun1_civil_3_open car_hatchback car_sedan

Vehicles

Name der Classe, Einheit	Name des Vehicle		Treffer Punkte	Cargo Plätze	Besatzung
Landeinheiten West					
M1Abrams	M1A1		900	0	3
M113	M113		150	9	2
M113Ambul	M113 Krankentransporter		150	3	1
Vulcan	M163 Flugabwehr		150	0	2
HMMWV	HMMWV		70	3	1

HMMWV50	HMMWV mit M2		70	3	2
HMMWVTOW	HMMWV mit TOW		70	3	2
HMMWVMK	HMMWV mit MK19		70	3	2
Truck5tMG	Truck 5t mit M2		20	12	2
Truck5t	Truck 5t		20	13	1
Truck5tOpen	Truck 5t ohne Plane		20	13	1
Truck5tRepair	Truck 5t Instandsetzung		20	1	1
Truck5tReammo	Truck 5t Munition		20	1	1

Truck5tRefuel	Truck 5t Treibstoff		20	1	1
Stryker_ICV_M2	Stryker mit M2		200	9	2
Stryker_TOW	Stryker mit TOW		200	0	2
Stryker_MK	Stryker mit MK19		200	9	2
M1030	M1030		50	1	1
Lufteinheiten West					
AH6	AH6		30	1	1
MH6	MH6		30	5	1
AH1W	AH1Z Cobra		60	0	2

UH60	UH60 mit FFAR		40	13	1
UH60MG	UH60 mit 2x MG		40	13	3
AV8B	AV8B mit GBU		60	0	1
	AV8B mit Sidwinder		60	0	1
A10	A 10 mit Maverik		85	0	1
Camel	Camel		15	0	1
ParachuteWest	Fallschirm		NC	0	1
Wassereinheiten West					
Zodiac	CRRC		20	4	1
RHIB	RHIB Transport		30	7	3

RHIB2Turret	RHIB mit M2 und MK19		30	7	3
Landeinheiten Widerstand					
M113_RACS	M113 (RACS)		150	11	2
Vulcan_RACS	Vulcan (RACS) Flugabwehr		150	2	2
Landrover_Closed	4x4		20	6	1
Landrover	4x4 ohne Plane		20	6	1
LandroverMG	4x4 mit M2		20	1	2
Lufteinheiten Widerstand					
AH6_RACS	AH6		30	1	1

MH6_RACS	MH6		30	5	1
Parachute	Fallschirm		NC	0	1
Wassereinheiten Widerstand					
Zodiac2	CRRC		20	4	1
Landeinheiten east					
T72	T72		750	0	3
BMP2	BMP2		250	7	3
BMP2Ambul	BMP2 Krankentransporter		250	3	1
ZSU	ZSU Shilka Luftabwehr		250	0	3
UAZ	UAZ		20	6	1
UAZMG	UAZ		20	1	2

UAZ_AGS30	UAZ		20	1	2
Ural	Ural		20	14	1
UralOpen	Ural ohne Plane		20	14	1
UralRefuel	Ural Treibstoff		20	2	1
UralReammo	Ural Munition		20	2	1
UralRepair	Ural Instandsetzung		20	2	1
TT650G	TT650G		50	1	1
BRDM2	BRDM 2		120	5	2
BRDM2_ATGM	BRDM2 mit ATGM		120	3	2
Lufteinheiten east					

KA50	KA50		60	0	1
Mi17	Mi17 mit Raketen		40	16	1
Mi17_MG	Mi17 mit MG		40	16	2
SU34	SU34 mit AAM		50	0	2
SU34B	SU34 mit AGM		50	0	2
Camel2	CameleE		15	0	1
ParachuteEast	Fallschirm		NC	0	1
Wassereinheiten east					
PBX	PBX Transporter		20	3	1
Landeinheiten					
TT650C	TT650		50	1	1

UralCivil	Ural (Civil)		20	14	1
UralCivil2	Ural (Civil 2)		20	14	1
Skoda	Skoda weiß		20	3	1
SkodaBlue	Skoda blau		20	3	1
SkodaRed	Skoda rot		20	3	1
SkodaGreen	Skoda grün		20	3	1
datsun1_civil_1_open	Pick-Up		20	3	1
datsun1_civil_2_covered	Pick-Up 2		20	3	1
datsun1_civil_3_open	Pick-Up 3		20	1	1

hilux1_civil_1_open	Geländewagen		20	1	1
hilux1_civil_2_covered	Geländewagen 2		20	1	1
hilux1_civil_3_open	Geländewagen 3		20	1	1
car_sedan	Sedan		20	3	1
car_hatchback	Hatchback		20	3	1
tractor	Trecker		20	0	1
Bus_city	Bus		20	12	1

Skript Befehle

<p>!_a nicht die Lokale Variable _a.</p>	<p>Skript Example</p> <pre>_a = alive John; if(!_a) then { hint" John ist tot!"; } Ist das selbe wie. _a = alive John;</pre>
--	--

	<pre>if not(_a) then { hint "John ist tot!"; }</pre>
<pre>!=</pre> <p>Abfrage a ungleich b</p>	<p>Skript Example</p> <pre>side John != west oder nameofplayer != "John Done" oder group player != group grp1 oder if (fuel truck1 != 0.5) then { hint "der Tank ist halb leer" }</pre>
<pre>a % b</pre> <p>Resultat von a durch b</p>	<pre>12 % 4</pre> <p>Ergebnis 3</p>
<pre>a && b</pre> <p>und Verknüpfung</p>	<pre>if (not alive a) && (not alive b) then {hint "a und b sind tot"}</pre>
<pre>a<b a>b</pre> <p>Vergleiche</p>	<pre>if (fuel truck1 < 0.5) then {hint "Treibstoff ist 50%."}</pre>
<pre>a<=b a>=b</pre> <p>Vergleich oder gleich</p>	<pre>if (fuel truck1 <= 0.5) then {hint "Treibstoff ist 50%."}</pre>
<pre>*</pre> <p>multipliziert a und b</p>	<pre>_wert = 10 * 2</pre> <p>Ergebnis 20</p>
<pre>+</pre> <p>addieren</p>	<p>Skript Example</p> <pre>_a = 2 + 3</pre> <p>Ergebnis 5</p> <p>Skript Example</p> <p>Strink concatenation - "hallo" + "ihr" + "spieler"</p> <p>Ergebnis Strink "hallo ihr spieler"</p>
<pre>-</pre> <p>subtrahieren</p>	<pre>_a = 5 - 2</pre> <p>Ergebnis 3</p> <pre>_a= 5 - 2</pre> <p>Ergebnis 7</p>
<pre>/</pre> <p>dividieren</p>	<pre>_a = 10 / 2</pre> <p>Ergebnis 5</p>
<pre>==</pre> <p>Vergleichen</p>	<pre>if (John == leader group grp1) then { hint "John ist der Chef"" } else { hint "John ist nicht der Chef" }</pre>
<pre>a ^ b</pre> <p>Exponent</p>	<pre>3 ^ 4 = 3 * 3 * 3 * 3</pre>
<pre>or</pre> <p>oder Abfrage</p>	<pre>a b ist gleich a or b</pre>
<pre>=</pre>	<pre>_a = 5</pre>

Gleichsetzen	<code>_mygroup = group player</code> <code>_enemytank = mytank 1</code>
abs	<code>_x = abs -5</code> Ergebnis 5
accTime	<code>_a = accTime</code>
acos Cosinus	<code>_cos = acos 0.6</code> Ergebnis 60
action Aktion einfügen	<code>unit action action</code> <code>John action ["eject", vehicle John]</code>
actionKeys Symbole Text für laden etc	<code>actionKeys action</code> <code>array = actionKeys</code> <code>"ReloadMagazine"</code>
actionKeysImages [action, maxKeys] Symbole Text für laden etc	<code>actionKeysImages action</code> <code>text = actionKeysImages</code> <code>"ReloadMagazine"</code>
actionKeysNames [action, maxKeys]	<code>actionKeysNames action</code> <code>list = actionKeysNames</code> <code>"ReloadMagazine"</code>
activateAddons [addon1, ...] Schaltet addons ein.	<code>activateAddons ["BISOFP"]</code>
activateKey keyname Aktiviert zB. eine neue Mission.	<code>activateKey "M04"</code>
unit addaction [action, script,(arguments,priority, showWindow, hideOnUse, shortcut)] or unit addAction [title, filename]	<code>genAct = generator addAction</code> <code>["Switch on generator",</code> <code>"activate_generator.sqs"]</code>
unit addMagazine magazineName fügt der Einheit ein Mag zu.	<code>Alex addMagazine</code> <code>"30Rnd_556x45_Stanag"</code>
vehicle addMagazineCargo [magazineName, count] fügt der Einheit im Laderaum Magaziene zu.	<code>LKW addMagazineCargo</code> <code>["30Rnd_556x45_Stanag",5]</code>
addMagazinePool [magazineName, count] fügt dem Pool der Campaigns Magaziene zu.	<code>addMagazinePool ["M16",20]</code>
unit addRating rating gibt der unit Punkte	<code>addRating 2000</code>
unit addScore score fügt der Abschussliste Punkte zu	<code>player addScore 10</code>
addSwitchableUnit person Schaltet die unit frei, diese ist dann spielbar.	<code>addSwitchableUnit player</code>
group addWaypoint [center, radius] Gibt der Gruppe einen neuen Wegpunkt	<code>wp = grp addWaypoint [position</code> <code>player,0]</code>
unit addWeapon weaponName Gibt der unit eine Waffe	<code>player addWeapon "M4AIM"</code>
truck addWeaponCargo ["M4AIM",5] fügt der Einheit im Laderaum Waffen zu.	<code>LKW addWeaponCargo</code> <code>["M4AIM",5]</code>
addWeaponPool [weaponName, count] fügt dem Pool der Campaigns Waffen zu.	<code>addWeaponPool ["M4AIM",5]</code>
alive bedarf keiner Erklärung	<code>?!(alive player) : exit</code>
unit allowFleeing courage	<code>group1 allowFleeing 1</code>

Set the courage of the group or unit.	
unitArray allowGetIn allow Set if the units given in the list are allowed to enter vehicles.	[soldier1, soldier2] allowGetIn true
unit ammo weaponName Muni wird wieder hoch gesetzt.	player ammo "M4AIM"
a and b und Verknüpfung	if (alive player) and (enemycount==0) then {hint "you win"}
object animate [animation, phase] zB. für öffnen und schließen von Türen	building animate ["maindoor", 1]
object animationPhase animation Return the phase of the given animation on the given object, which is set by the animate command.	building animate ["maindoor", 1] ~2 _p = building animationPhase "maindoor"
animationState man Return the name of the current primary animation.	_state = animationState player
asin x Sinus	asin 0.5 Ergebnis 30
assert condition Tests a condition and if the condition is false, halts the program.	assert (_x>0)
unit assignAsCargo vehicle Zählt die unit dem Speicher eines vehicle zu.	soldier1 assignAsCargo truck [soldier1] orderGetIn true
unit assignAsCommander vehicle Zählt die unit dem Speicher eines vehicle zu.	soldier1 assignAsCommander tank [soldier1] orderGetIn true
unit assignAsDriver vehicle Zählt die unit dem Speicher eines vehicle zu.	soldier1 assignAsDriver tank [soldier1] orderGetIn true
unit assignAsGunner vehicle Zählt die unit dem Speicher eines vehicle zu.	soldier1 assignAsGunner tank [soldier1] orderGetIn true
assignedTarget vehicle Bestimmt das Ziel der Einheit	_target = assignedTarget T72
vehicle assignTeam team Assigns the unit or, in case of a vehicle, its commander unit to the given team. The possible team values are: "MAIN", "RED", "GREEN", "BLUE" and "YELLOW".	soldier2 assignTeam "RED"
atan Tangens	atan 1 Ergebnis 45
x atan2 y Winkel zum Vektor X/Y	5 atan2 3 Ergebnis 59.0362
atg Gegenstück zum atan	
attackEnabled group Return whether leader can issue attack commands.	if (not attackEnabled soldier) then {soldier setCombatMode "Careless"}
behaviour unit Verhalten der Einheit	soldier setBehaviour "CARELESS" _b = behaviour soldier
benchmark 3D Performance	? (benchmark>2000) : setViewDistance 2000

boundingBox object Returns a bounding box of given object in model coordinate space.	_box = boundingBox abrams
building buildingPos index Gibt die Position des Gebäudes	soldier setPos (buildingPos [house1, 2]) soldier setPos (house1 buildingPos 2)
buttonAction idc Return the action assigned to a control of the currently active user dialog.	buttonSetAction [100, {player exec "reply.sqs"}] _action = buttonAction 100
buttonAction control Returns the action assigned to the given button or active text.	_action = buttonAction _button
buttonSetAction [idc, code] Set the action of a control of the currently active user dialog.	buttonSetAction [100, {player exec "reply.sqs"}]
buttonSetAction control Assign action to control with id idc of topmost user dialog.	buttonSetAction [100,"player exec""reply.sqs"""]
cadetMode Returns if the player is currently playing in cadet or veteran mode.	? (cadetMode) : ai_soldier setSkill 0.1
pars call body Executes the function body.	_fAdd = compile loadFile "add.sqf" [1,2] call _fAdd
call code Executes the given code.	call {"x = 3"} or _n = 3; call compile format ["var%1 = 0",_n];
camera camCommand command Executes a command on the given camera / actor object.	_camera camCommand "manual on"
camera camCommit time Smoothly conduct the changes that were assigned to a camera within the given time. If the time is set to zero, the changes are done immediately.	; create a camera object _cam = "camera" camCreate [5600,4800,10] _cam camSetTarget player _cam cameraEffect ["internal", "BACK"] _cam camCommit 0 ; smoothly move the camera to its new position in 6 seconds _cam camSetPos [5680,4720,20] _cam camCommit 6 @camCommitted _cam ; proceed
camera camCommitPrepared time Smoothly conduct the changes that were assigned to a camera within the given time. If the time is set to zero, the changes are done immediately.	; create a camera object _cam = "camera" camCreate [5600,4800,10] _cam camPrepareTarget player _cam cameraEffect ["internal",

	<pre>"BACK"] _cam camCommitPrepared 0 ; smoothly move the camera to its new position in 6 seconds _cam camPreparePos [5680,4720,20] _cam camCommitPrepared 6 @camCommitted _cam ; proceed</pre>
<p>camCommitted camera Checks if the conduction of the last camCommit call already finished.</p>	<pre>; create a camera object _cam = "camera" camCreate [5600,4800,10] _cam camSetTarget player _cam cameraEffect ["internal", "BACK"] _cam camCommit 0 ; smoothly move the camera to its new position in 6 seconds _cam camSetPos [5680,4720,20] _cam camCommit 6 @camCommitted _cam ; proceed</pre>
<p>type camCreate position Create a camera or a seagull object on the given position.</p>	<pre>_cam = "camera" camCreate (position player)</pre>
<p>camDestroy object Destroy an object created with camCreate.camDestroy is conducted immediately, the command doesn't wait for camCommit.</p>	<pre>camDestroy _cam</pre>
<p>camera cameraEffect [type, position] Switch to the given camera or object with the given effect.</p>	<pre>_cam cameraEffect ["internal", "BACK"]</pre>
<p>cameraOn Returns the vehicle to which the camera is attached.</p>	<pre>_a = cameraOn</pre>
<p>camera camPreload time Preload the scene for the prepared camera. Time gives timeout, zero means no (infinite) timeout.</p>	<pre>_camera camPreload 5</pre>
<p>camPreloaded _camera Checks whether the camera has finished preloading.</p>	<pre>camPreloaded _camera</pre>
<p>camera camPrepareBank bank Prepares the camera bank angle.</p>	<pre>_camera camPrepareBank -0.1</pre>
<p>camera camPrepareDir direction Prepares the camera heading.</p>	<pre>_camera camPrepareDir 150</pre>
<p>camera 'camPrepareDive dive Prepares the camera dive angle.</p>	<pre>_camera camPrepareDive -0.1</pre>

camera camPrepareFocus focusRange focusRange is in format [distance,blur]. Prepares the camera focus blur. [-1,1] will reset default values (auto focusing), [-1,-1] will disable postprocessing (all is focused).	_camera camPrepareFocus [50, 1]
camera camPrepareFov fieldOfView Prepares the camera field of view (zoom).	_camera camPrepareFov 0.1
camera camPrepareFovRange fovRange Prepares the camera field of view range for auto zooming.	_camera camPrepareFovRange [0.1, 0.5]
camera camPreparePos position Prepares the camera position (format Position).	_camera camPreparePos getPos player
camera camPrepareRelPos position Prepares the camera position relative to the current position of the current target (see camPrepareTarget).	_camera camPrepareRelPos [10,5]
camera camPrepareTarget target Prepares the camera target to a position or to a target.	_camera camPrepareTarget getPos player or _camera camPrepareTarget player
camera camSetBank bank Set camera bank angle. Does not commit changes.	_camera camSetBank -0.1
camera camSetDir direction Set the direction of the given camera.Needs the call of camCommit to be conducted.	_cam camSetDir 90
camera camSetDive dive Set camera dive angle. Does not commit changes.	_camera camSetDive -0.1
camera camSetFocus focusRange focusRange is in format [distance,blur]. Sets the camera focus blur. It does not automatically commit changes.	_camera camSetFocus [50, 1]
camera camSetFov level Set the zoom level (field of view) of the given camera. The default zoom level is 0.7, 0.01 is the nearest and 2 the farthest zoom value. The angle of the field of view is the FOV*120 degrees.Needs the call of camCommit to be conducted.	_cam camSetFov 0.7
camera camSetFovRange [start,end] Set the zoom level (field of view) start and end values for automatic zooming. The default zoom level is 0.7, 0 is the nearest and 1 the most far zoom value.Needs the call of camCommit to be conducted.	_cam camSetFovRange [0.1,0.7]
camera camSetPos position Set the position of the given camera or seagull.Needs the call of camCommit to be conducted.	_cam camSetPos [2300,1000,130]
camera camSetRelPos position Set the position of the given camera relative to its target set with camSetTarget.Needs the call of	_cam camSetTarget _car _cam camSetRelPos [0,10,8]

camCommit to be conducted.	
camera camSetTarget target Set the target object or position where the given camera should point at. Needs the call of camCommit to be conducted.	_camera camSetTarget player or _camera camSetTarget [2540,1503,26]
camUseNVG set Setzt das NVG vor die Kamera.	
canFire vehicle Returns if the given vehicle is still able to fire. This command checks only the damage value, not the ammo!	?!(canFire tank) : player sideChat "Tank disabled!"
canMove vehicle Returns if the given vehicle is still able to move. This command checks only the damage value, not the amount of fuel!	?!(canMove tank) : player sideChat "He's nailed on the ground! Now hurry!"
canStand soldier Returns if the given soldier is able to stand up.	?!(canStand player) : player groupChat "My legs! They hit my legs!"
captive unit Macht die Unit für den Feind zum Freund.	?!(captive general) : general setCaptive true
case b See switch do	
try-block catch code Processes code when an exception is thrown in a try block. The exception caught can be found in the _exception variable.	
ceil x The ceil value of x.	ceil 5.25 Ergebnis 6 or ceil -5.25 Ergebnis -5
cheatsEnabled Checks whether cheats are enabled (whether the designers' version is running).	
civilian Pre-defined variable for the civilian side. When used in a format statement (hint format["%1",civilian]), the string returned is "CIV".	?((side _unit)==civilian) : hint "This is a civilian unit!"
clearMagazineCargo vehicle Löscht alle mags im vehicle	clearMagazineCargo truck
clearMagazinePool Löscht alle Mags im Pool der Campaigne	
clearRadio Clean up the content of radio protocol history.	
clearWeaponCargo vehicle Löscht alle Waffen im vehicle	clearWeaponCargo truck
clearWeaponPool Löscht alle Waffen im Pool der Campaigne	
closeDialog idc	closeDialog 0

Close the currently active user dialog.	
display closeDisplay exitcode Close given display.	_display closeDisplay IDC_OK
combatMode group Returns the combat mode of the given unit. See setCombatMode for more information about combat modes.	?((combatMode grp1)=="BLUE") : grp1 setCombatMode "GREEN"
commander vehicle Liefert den Namen des commander eines vehicle.	(commander tank) action ["getout",tank]
unit commandFire target Order the given unit to fire on the given target (via the radio). If the target is objNull, the unit is ordered to fire on its current target (set with doTarget or commandTarget).	ESoldier1 commandFire WSoldier1
unit commandFollow followedunit Order the given unit to follow the given other unit (via the radio).	soldier1 commandFollow soldier2
unit(s) commandFSM [fsm name, position, target] Orders a unit to process command defined by FSM file (via the radio).	soldierOne commandFSM ["move.fsm", position player, player]
commandGetOut unit Den Commander steigt aus.	commandGetOut unitOne
unit commandMove position Oder sich zu dem Punkt zu begeben.	soldier1 commandMove (position "Marker1")
commandStop unit Order die Gruppe zu stoppen.	commandStop soldier1
unit commandTarget target Weißt ein Ziel zu.	ESoldier1 commandTarget WSoldier1
unit commandWatch target Weißt eine Richtung zu die beobachtet wird.	soldierOne commandWatch markerPos "MarkerMoveOne" or [s1,s2] commandWatch player
comment comment Define a comment. Mainly used in SQF Syntax, as you're able to introduce comment lines with semicolons in a SQS script.	***.sqf comment "This is a commented line" or ***.sqf comment "This is a commented line";
compile expression Compile expression.	_function = "a = a + 1"; _compiled = compile _function; call _compiled;
composeText [text1, text2, ...] Creates a structured text by joining the given structured or plain texts.	txt = composeText ["First line", image "data\isniper.paa", lineBreak, "Second line"]
cos x Cosinus einer Zahl Winkel	cos 60 Ergebnis 0.5
condition count array Counts elements in array for which given condition is true. It is calculated as follows: Set count to 0. For each element of array assign element as _x and evaluate condition expression. If true increase count.	_array = [1,2,2,3,4,4,4,5,6] {_x==2} count _array Ergebnis 2

count array Count number of elements in the array.	count [0, 0, 1, 2] Ergebnis 4 count units groupOne count thislist
count configname Returns count of subentries.	_count = count (configFile >> "CfgVehicles")
unit countEnemy array Count how many units in the array are considered enemy to the given unit.	player countEnemy list triggerOne
unit countFriendly array Count how many units in the array are considered friendly to the given unit.	player countFriendly list triggerOne
side countSide array Count how many units in the array belong to given side.	west countSide list triggerOne
type countType array Count how many units in the array are of given type. Other than typeOf this command also works with parent classes like "Air", "Tank" and "Car". For a full class reference see Classes.	"Tank" countType list triggerOne
unit countUnknown array Count how many units in the array are unknown to the given unit.	player countUnknown list triggerOne
createAgent [type, position, markers, placement, special] Creates an (independent) agent (person) of the given type (type is a name of a subclass of CfgVehicles). If the markers array contains several marker names, the position of a random one is used, otherwise, the given position is used. The unit is placed inside a circle with this position as its center and placement as its radius. Special properties can be: "NONE" and "FORM".	agent = createAgent ["SoldierWB", position player, [], 0, "FORM"]
createCenter side Creates a new AI center for the given side. A 'center' is something each side needs to have to be able to communicate. By default, all centers for units which are present in the mission are created before the mission is started. This command can be used to initialize a side which has no units present in the Mission.sqm, so that you can spawn groups and units for it.	center = createCenter east
createDialog className Create a dialog which is defined either in the mission's description.ext, in the campaign's description.ext or in the global resource.cpp. The given name has to be the class name used in one of these files. If already another dialog is opened, the desired dialog is created as child dialog of the already opened one.	_ok = createDialog "RscDisplayGame" ?!(_ok) : hint "Dialog couldn't be opened!"
parent createDisplay name	_display createDisplay

Create child display of given display and load from resource "name".	"RscObserver"
createGroup side Creates a new AI group for the given center (side). The center must already be in the Mission.sqm or have been initialized with createCenter.	group = createGroup east
createGuardedPoint [side, position, idStatic, vehicle] Adds a point guarded by the given side. If idStatic is not negative, the position of a static object with the given id is guarded. If the given vehicle is valid, the starting position of the vehicle is guarded, otherwise the given position is guarded.	point = createGuardedPoint [east, [0, 0], -1, vehicle player]
createMarker [name, position] Creates a new marker at the given position. The marker name has to be unique.	marker= createMarker [Marker1, position player]
createMarkerLocal [name, position] Creates a new marker at the given position. The marker name has to be unique.	marker= createMarkerLocal [Marker1, position player]
createMine [type, position, markers, placement] Creates a mine of the given type (type is the name of the subclass of CfgVehicles). If the markers array contains several marker names, the position of a random one is used, otherwise, the given position is used. The mine is placed inside a circle with this position as its center and placement as its radius.	mine = createMine ["MineMine", position player, [], 0]
createSoundSource [type, position, markers, placement] Creates a sound source of the given type (type is the name of the subclass of CfgVehicles). If the markers array contains several marker names, the position of a random one is used, otherwise, the given position is used. The sound source is placed inside a circle with this position as its center and placement as its radius.	soundSource = createSoundSource ["LittleDog", position player, [], 0]
obj createTarget [type, position, typeAccuracy, posAccuracy] Create a target. Not yet functioning.	
createTrigger [type, position] Creates a new trigger on the given position. An object of the given type is created; this type must be a class name in CfgNonAI Vehicles or CfgVehicles with simulation = detector. An array containing all units that have activated the trigger is available via list triggerobj.	trgobj = createTrigger ["EmptyDetector", position player] _trgunits = list trgobj
type createUnit [position, group, init*, skill*, rank*] * optional, if one is set, all precedent arguments have to be set too	"SoldierWB" createUnit [position player, group player]

Create an interacting AI soldier.	
group createUnit [type, position, markers, placement, special] Creates a unit (person) of the given type (type is a name of a subclass of CfgVehicles) and makes it a member of the given group. If the markers array contains several marker names, the position of a random one is used. Otherwise, the given position is used. The unit is placed inside a circle with this position as its center and placement as its radius. Special properties can be: "NONE" and "FORM".	unit = group player createUnit ["SoldierWB", position player, [], 0, "FORM"]
type createVehicle position Create an empty vehicle/object of given type on given position. For a full class reference see Classes.	_jeep = "Jeep" createVehicle (position player)
createVehicle [type, position, markers, placement, special] Creates a vehicle of the given type (type is the name of the subclass in CfgVehicles). If the markers array contains several marker names, the position of a random one is used. Otherwise, the given position is used. The vehicle is placed inside a circle with this position as center and placement as its radius. Special properties can be: "NONE", "FLY" and "FORM".	veh = createVehicle ["ah1w", position player, [], 0, "FLY"]
type createVehicleLocal pos Creates an empty vehicle of the given type. See CfgVehicles for possible type values. Vehicle is not transferred through network in MP games.	_tank = "M1Abrams" createVehicleLocal getMarkerPos "tankFactory"
crew vehicle Returns the crew of the given vehicle.	player in (crew tank)
ctrlActivate control Launch actions attached to given (button based) control.	ctrlActivate _control
control ctrlCommit time Commit control animation.	_control ctrlCommit 2
ctrlCommitted control Check if the control animation is finished.	_done = ctrlCommitted _control
ctrlEnable [idc, enable] Enable or disable a control of the currently active user dialog. Disabled controls cannot be clicked onto. Read Dialog Control for more information about user dialogs and controls.	ctrlEnable [100,false]
control ctrlEnable enable Enable / disable the control with id idc of topmost user dialog.	ctrlEnable [100, false]
ctrlEnabled idc Returns if a control on the currently active user dialog is enabled. Disabled controls cannot be clicked onto. Read Dialog Control for more information about user dialogs and controls.	?!(ctrlEnabled 100) : ctrlEnable [100,true]

ctrlEnabled control Return if control with id <code>idc</code> of topmost user dialog is enabled.	<code>_enabled = ctrlEnabled 100</code>
ctrlFade control Returns the current fade factor of control.	<code>_scale = ctrlFade _control</code>
map ctrlMapAnimAdd frame Adds the next frame to the map animation. The format of frame is [time, zoom, position], the format of position is Position2D.	<code>_map ctrlMapAnimAdd [1, 0.1, getMarkerPos "anim1"]</code>
ctrlMapAnimClear control Clears the map animation.	
ctrlMapAnimCommit control Plays the map animation.	
ctrlMapAnimDone control Checks whether the map animation has finished.	
ctrlMapScale control Return the current scale of the map control.	
ctrlParent control Returns container of the given control.	<code>_display = ctrlParent _control</code>
ctrlPosition control Returns the current position of control as [x, y] array.	<code>_pos = ctrlPosition _control</code>
ctrlScale control Returns the current scale of the control.	<code>_scale = ctrlScale _control</code>
display ctrlSetActiveColor color Sets text color of given control when control is selected.	<code>_control ctrlSetActiveColor [1, 0, 0, 1]</code>
display ctrlSetBackgroundColor color Sets background color of given control.	<code>_control ctrlSetBackgroundColor [1, 0, 0, 1]</code>
control ctrlSetEventHandler [handler name, function] Sets given event handler of given control.	<code>_control ctrlSetEventHandler ["KeyDown", ""]</code>
control ctrlSetFade fade Sets wanted transparency for control animation.	<code>_control ctrlSetFade 1</code>
ctrlSetFocus control Set the input focus on given control.	<code>ctrlSetFocus _control</code>
control ctrlSetFont name Sets the main font of given control.	<code>_control ctrlSetFont "TahomaB"</code>
control ctrlSetFontH1 name Sets H1 font of given HTML control.	<code>_control ctrlSetFontH1 "TahomaB"</code>
control ctrlSetFontH1B name Sets H1 bold font of given HTML control.	<code>_control ctrlSetFontH1B "TahomaB"</code>
control ctrlSetFontH2 name Sets H2 font of given HTML control	<code>_control ctrlSetFontH2 "TahomaB"</code>
control ctrlSetFontH2B name Sets H2 bold font of given HTML control.	<code>_control ctrlSetFontH2B "TahomaB"</code>
control ctrlSetFontH3 name Sets H3 font of given HTML control.	<code>_control ctrlSetFontH3 "TahomaB"</code>
control ctrlSetFontH3B name Sets H3 bold font of given HTML control.	<code>_control ctrlSetFontH3B "TahomaB"</code>

control ctrlSetFontH4 name Sets H4 font of given HTML control.	_control ctrlSetFontH4 "TahomaB"
control ctrlSetFontH4B name Sets H4 bold font of given HTML control.	_control ctrlSetFontH4B "TahomaB"
control ctrlSetFontH5 name Sets H5 font of given HTML control.	_control ctrlSetFontH5 "TahomaB"
control ctrlSetFontH5B name Sets H5 bold font of given HTML control.	_control ctrlSetFontH5B "TahomaB"
control ctrlSetFontH6 name Sets H6 font of given HTML control.	_control ctrlSetFontH6 "TahomaB"
control ctrlSetFontH6B name Sets H6 bold font of given HTML control.	_control ctrlSetFontH6B "TahomaB"
control ctrlSetFontHeight height Sets the main font size of given control.	_control ctrlSetFontHeight 0.05
control ctrlSetFontHeightH1 height Sets H1 font size of given HTML control.	_control ctrlSetFontHeightH1 0.05
control ctrlSetFontHeightH2 height Sets H2 font size of given HTML control.	_control ctrlSetFontHeightH2 0.05
control ctrlSetFontHeightH3 height Sets H3 font size of given HTML control.	_control ctrlSetFontHeightH3 0.05
control ctrlSetFontHeightH4 height Sets H3 font size of given HTML control.	_control ctrlSetFontHeightH4 0.05
control ctrlSetFontHeightH5 height Sets H5 font size of given HTML control.	_control ctrlSetFontHeightH5 0.05
control ctrlSetFontHeightH6 height Sets H6 font size of given HTML control.	_control ctrlSetFontHeightH6 0.05
control ctrlSetFontP height Sets P font size of given HTML control.	_control ctrlSetFontP 0.05
control ctrlSetFontP name Sets P font of given HTML control.	_control ctrlSetFontP "TahomaB"
control ctrlSetFontPB name Sets P bold font of given HTML control.	_control ctrlSetFontPB "TahomaB"
display ctrlSetForegroundColor color Sets background color of given control. Color is in format Color.	_control ctrlSetForegroundColor [1, 0, 0, 1]
control ctrlSetPosition [x, y] Sets wanted position for control animation.	_control ctrlSetPosition [0.5, 0.5]
control ctrlSetScale scale Sets wanted scale for control animation.	_control ctrlScale 0.5
control ctrlSetStructuredText structured text Set the structured text which will be displayed in structured text control.	_control ctrlSetStructuredText parseText "First line
/>Second line"
ctrlSetText [idc, text] Set the text of a control of the currently active user dialog. This command can be used for static texts, buttons, edit lines and active texts as well as for images, where you can use it to set the image path. Read Dialog Control for more information about user dialogs and controls.	ctrlSetText [100, "Hello world"]

control ctrlSetText text Sets the text that will be shown in given control.	_control ctrlSetText "Hello, world."
display ctrlSetTextColor color Sets text color of given control. Color is in format Color.	_control ctrlSetTextColor [1, 0, 0, 1]
display ctrlSetTooltip text Sets tooltip text of given control.	_control ctrlSetTooltip "tooltip" or (findDisplay 10000) displayCtrl 10001 ctrlSetTooltip "ThisIsAGoodTip"
display ctrlSetTooltipColorBox color Sets tooltip border color of given control. Color is in format Color.	_control ctrlSetTooltipColorBox [1, 0, 0, 1]
display ctrlSetTooltipColorShade color Sets tooltip background color of given control. Color is in format Color.	_control ctrlSetTooltipColorShade [1, 0, 0, 1]
display ctrlSetTooltipColorText color Sets tooltip text color of given control. Color is in format Color.	_control ctrlSetTooltipColorText [1, 0, 0, 1]
ctrlShow [idc, show] Set if a control of the currently active user dialog is shown or not. Read Dialog Control for more information about user dialogs and controls.	ctrlShow [100,false]
control ctrlShow show Show / hide given control.	_control ctrlShow false
ctrlText idc Return the text of a control of the currently active user dialog. This command can be used on static texts, buttons, edit lines and active texts as well as for images, where it returns the image path. Read Dialog Control for more information about user dialogs and controls.	ctrlText 100
ctrlText control Returns the text shown in given control.	_text = ctrlText _control
ctrlType Returns value representing type of control.	_type = ctrlType _control
ctrlVisible idc Returns if a control of the currently active user dialog is shown or not. Read Dialog Control for more information about user dialogs and controls.	ctrlVisible 100
currentCommand vehicle Return the current command type (empty string when no command) for the commander of given vehicle (or for a given soldier). Value returned can be one of: "WAIT", "ATTACK", "HIDE", "MOVE", "HEAL", "REPAIR", "REFUEL", "REARM", "SUPPORT", "JOIN", "GET IN", "FIRE", "GET OUT", "STOP", "EXPECT", "ACTION", "ATTACKFIRE",	
cutObj [name, type, speed*] * optional	cutObj ["TVSet", "PLAIN"]

Display an object defined in the mission's description.ext, the campaign's description.ext or the global resource.cpp.	cutObj ["TVSet","PLAIN",2]
cutRsc [name, type, speed*] * optional Display a resource defined in the mission's description.ext, the campaign's description.ext or the global resource.cpp.	cutRsc ["binocular","PLAIN"] cutRsc ["binocular","PLAIN",2]
cutText effect Text background - effect is in format ["name", "type", speed] or ["name", "type"]. If speed is not given, 1 is assumed. Type may be one of: "PLAIN" "PLAIN DOWN" "BLACK" "BLACK FADED" "BLACK OUT""BLACK IN" "WHITE OUT" "WHITE IN" See Title Effect Type for more information about these values.	cutText ["" ,"BLACK OUT"] cutText ["Hello World!","PLAIN",2]
damage object Return the damage value of an object.	? ((damage player)>0.1) : player groupChat "I'm hurt! Medic!"
date Return the actual mission date and time as an array [year, month, day, hour, minute].	
daytime Returns the current ingame time in hours.	; assume it is 16:30 _daytime = daytime Ergebnis 16.5
debugLog anything Dump argument type and value to debugging output. Note: this command has no real use in the retail version, because it does nothing.	debugLog player
default a see switch.	
deg x Convert a number from radians to degrees.	deg1 Ergebnis 57.295
deleteCenter side Destroys the AI center of the given side.	deleteCenter east
deleteCollection object Delete a collection.	
deleteGroup group Destroys the given AI group.	deleteGroup groupname
deleteIdentity identity Delete an identity (created with saveIdentity) from the campaign's progress file.	deleteIdentity "playerIdentity"
deleteMarker name Destroys the given marker.	deleteMarker "Marker1"
deleteMarkerLocal name Destroys the given marker.	deleteMarkerLocal "Marker1"
deleteStatus status Delete a status (created with saveStatus) from the campaign's progress file.	deleteStatus "playerStatus"
deleteTarget target Delete a target.	deleteTarget target1
deleteVehicle object	deleteVehicle tank

Delete an object. Note that only units inserted in the mission editor and units created during the game's progress can be deleted by this command. Island objects and player units can't be removed.	
deleteWaypoint waypoint Removes the waypoint.	deleteWaypoint [grp, 2]
dialog Returns if a user dialog is present. Read Dialog Control for more information about user dialogs and controls.	?!(dialog) : createDialog "Dialog1"
direction object Returns the direction an object is facing.	player setDir 90 _d = direction player
unit disableAI section Disable parts of the AI behaviour to get a better control over the actions of a unit. Possible values are: "TARGET" - stop the unit to watch the assigned target "AUTOTARGET" - prevent the unit from assigning a target independently and watching unknown objects "MOVE" - disable the AI's movement "ANIM" - disable ability of AI to change animation. Available only in Armed Assault. Note: All effects of disableAI command are cancelled after mission save or load. Note: In OFP is no way to undone this command	soldier1 disableAI "AUTOTARGET"
disableUserInput state Disable and enable the keyboard and mouse input, usually used during cutscenes. Be careful with the usage of this command, always remember to enable the user input again, as once the user input is disabled, you can only shut down OFP but not exit the mission with escape	disableUserInput true ; cutscene disableUserInput false
display displayCtrl idc Return child control with specified idc.	_control = _display displayCtrl 101
displayNull A non-existing display. This value is not equal to anything, including itself.	
display displaySetEventHandler [handler name, function] Sets given event handler of given display.	_control displaySetEventHandler ["KeyDown", ""]
dissolveTeam team Dissolves the given team. All members become members of the main team. Possible team values are: "RED", "GREEN", "BLUE" or "YELLOW".	dissolveTeam "RED"
obj1 or pos1 distance obj2 or pos2 Returns the distance in meters between two objects or positions.	?((player distance generator)<10) : player exec "activateGenerator.sqs"
while do code	while {a>b} do {a=a+1}

Repeat code while condition is true.	
switch do block Switch form.	switch (_a) do {case 1: {hint "1"}; case 2: {hint "2"}; default {hint "default"};} or switch (_a) do {case true: {hint "true"}; case false: {hint "false"}; default {hint "default"};} or switch (_a) do {case "string1": {hint "string1"}; case "string2": {hint "string2"}; default {hint "default"};}
orCommand do code End of for command, starts cycle.	for "_x" from 20 to 10 step -2 do {..code..}
unit doFire target Order the given unit(s) to fire on the given target (without radio messages). If the target is objNull, the unit is ordered to fire on its current target (set with doTarget or commandTarget).	ESoldier1 doFire WSoldier1
unit doFollow followedunit Order the given unit to follow the given other unit (without radio messages).	soldier1 doFollow soldier2
unit(s) doFSM [fsm name, position, target] Orders a unit to process command defined by FSM file (silently).	soldierOne doFSM' ["move.fsm", position player, player]
doGetOut unit Orders a unit to get out from the vehicle (silently).	doGetOut unitOne
unit doMove position Order the given unit(s) to move to the given position (without radio messages).	soldier1 doMove (position "Marker1")
doStop unit Order the given unit to stop (without radio messages).	doStop soldier1
unit doTarget target Order the given unit to target the given target (without radio messages).	ESoldier1 doTarget WSoldier1
unit doWatch position Order the given unit to watch the given position or target (without radio messages).	soldierOne doWatch markerPos "MarkerMoveOne" or soldierOne doWatch eastSoldier
map drawArrow [position1, position2, color] Draw an arrow on the map.	
map drawEllipse [center, a, b, angle, color, fill] Draw an ellipse on the map.	
map drawIcon [texture, color, position, width, height, angle, text, shadow] Draw an Icon on the map.	
map drawLine [position1, position2, color] Draw a line on the map.	

map drawRectangle [center, a, b, angle, color, fill] Draw a Rectangle on the map.	
driver vehicle Returns the driver of a vehicle.	(driver tank) action ["getout",tank]
drop array Creates a particle effect. This command is used to create smoke, fire and similar effects. The particles are single polygons with single textures that always face the player. They can be set to dynamically change their position, size, direction, can be set to different weights and more or less dependant on the wind.	drop ["cl_basic", "", "Billboard", 1, 1, [-3.5*(sin(direction xural)), -3.5*(cos(direction xural)),0], [random 0.1,random 0.1,random 0.5], 1, 0.005, 0.0042, 0.7, [0.3,3], [[0.5,0.5,0.5,0],[0.7,0.7,0.7,0.5],[0.9,0.9,0.9,0]], [0,1,0,1,0,1], 0.2, 0.2, "", "", xural]
east Pre-defined variable for the eastern side	?((side _unit)==east) : hint "This is an eastern unit!"
echo text Sends any text into the debugger console or the logfile. Present in internal version only, not working in the retail version.	echo "Text in logfile"
effectiveCommander vehicle Returns the effective commander (who really commands) of the vehicle.	
ifCode else elseCode Construct an array that can be processed by then.	if (a>b) then {c=0} else {c=1}
vehicle emptyPositions position Returns the number of given positions in the vehicle. Positions can be "Commander", "Driver", "Gunner" or "Cargo"	_freePositions = (vehicle player) emptyPositions "cargo"
unit enableAI section Enables parts of the AI behaviour. Section is one of: "TARGET" - enables watching assigned targets "AUTOTARGET" - enables independed target assigning and watching of unknown targets "MOVE" - enables movement. "ANIM" - enables animation.	soldierOne enableAI "Move"
group enableAttack enable Set if leader can issue attack commands.	group1 enableAttack true
enableEndDialog Enables the execution of a custom camera sequence after the players death, coded in the script onPlayerKilled.sqs.	
enableEnvironment enabled Enable/disable environmental effects (ambient life + sound).	enableEnvironment false
enableRadio state Enable and disable radio messages to be heard and shown in the left lower corner of the screen. That can be helpful during cutscenes.	enableRadio false
object enableReload enable	_vehicle enableReload false

Enable / disable reload right after magazine is empty.	
enableTeamSwitch enable Enable / disable team switch.	
enemy Enemy side (enemy to all units).	
vehicle engineOn state Activates and deactivates the engine of a vehicle.	?!(isEngineOn jeep) : jeep engineOn true
estimatedTimeLeft time Sets the "time left" value that is shown in the "Game in Progress" screen during multiplayer sessions. This command works in MP only. If you set param1, param2 or following in the mission's description.ext, you can use those values instead of time.	?("OBJ1" objectStatus "DONE") : estimatedTimeLeft 600
argument exec fileName Execute a script. The argument is passed to the script in the "_this" variable. Learn more about scripts under Scripts.	[player, jeep] exec "getin.sqs"
argument execVM filename Compile and execute SQF Script. Argument is passed to the script as local variable _this. The Script is first searched for in the mission folder, then in the campaign scripts folder and finally in the global scripts folder.	var = player execVM "test.sqf"
filename: String Compile and execute script (sqf). Script is compiled every time you use this command. The script is first searched for in the mission folder, then in the campaign scripts folder and finally in the global scripts folder.	var = execVM "test.sqf"
exit Stops the execution of a SQS script. Is ignored in SQF Scripts, use exitWith instead.	
if exitWith code If the result of condition is true, code is evaluated, and current block with result of code	if (_x>5) exitWith {echo "_x is too big"; _x}
expectedDestination person Return expected destination of unit as an array with format: [position, planningMode, forceReplan].	
exportLandscapeXYZ filename Exports landscape as XYZ file. Note: Not available in Retail Version	exportLandscapeXYZ myLandscape
time fadeMusic volume Changes the music volume smoothly within the given time.	7 fadeMusic 0
time fadeRadio volume Causes a smooth change in the radio volume. The change duration is given by time, the target	11 fadeRadio 0.1

volume by volume. The default radio volume is 1.0.	
time fadeSound volume Changes the sound volume smoothly within the given time.	4 fadeSound 0
false Always false.	
fillWeaponsFromPool unit Adds magazines from the campaign pool to the given unit, depending on his weapons.	fillWeaponsFromPool soldier
array find x Returns the position of the first array element that matches x, returns -1 if not found. Test is case-sensitive.	[0, 1, 2] find 1 result is 1 or if ((magazines player) find "Strela" > -1) then {hint "I got a Strela"}
object findCover [position, hidePosition, maxDist] Returns the object where the object should search for cover.	
findDisplay idd Find display by its IDD (which is defined in the description.ext or config).	_display = findDisplay 1
object findNearestEnemy position Find the nearest enemy from the specified position.	
finishMissionInit Finish world initialization before mission is launched.	
finite x True, if number is finite (not infinite and a valid number)	finite 10/0 result is false
unit fire weapon Forces a unit to fire the given weapon.	soldier fire "M16" or soldier fire ["throw","SmokeShell","SmokeShell"]
flag unit Returns the flag the unit is carrying.	_flag = flag soldier
flagOwner anyFlag Returns the owner of a flag.	_person = flagowner flagOne
fleeing unit Checks if a unit is fleeing.	? (fleeing east_unit) : player sideChat "We have won!"
floor Returns the next lowest integer in relation to x.	floor 5.25 Ergebnis 5 or floor -5.25 Ergebnis -6
aircraft flyInHeight altitude Sets the flying altitude for aircraft relatively to the ground surface. Avoid too low altitudes, as helicopters and planes won't evade trees and	helicopter flyInHeight 20

obstacles on the ground. The default flying altitude is 50 meters.	
fog Return the current fog.	
fogForecast Return the fog forecast.	
for forspec creates cycle, using C like style. See example.	for [{_x=1},{_x<=10},{_x=_x+1}] do {debugLog _x;}
for var Starts for sequence, use in complete form (see example).	for "_i" from 1 to 10 do {debugLog _i;}
forceEnd Enforces mission termination.	? (territory_lost) : forceEnd
forceMap state Displays the map on the screen during a mission.	forceMap true
object forceSpeed speed Force the speed of the given object.	
command forEach array Executes the given command on every item of an array. The array items are represented by _x. All commands are executed within one frame, so this command might cause performance loss when used on very large arrays or with very complex commands.	_x setdamage 1 } forEach units group player or {player addMagazine "M16"} forEach [1,2,3,4]
format [string, var1, var2 ...] Composes a string containing other variables or other variable types. Converts any variable type to a string. If you want to convert a string back to a number, use call. The array used with this command has to contain at least two items.	format ["Player side: %1 - Human players on that side: %2", side player, playersNumber side player]
formation grp Returns the current formation of a group.	formation group player
formationDirection person Return the direction unit watching in formation.	
formationLeader person Return leader of the formation.	
formationMembers person Return list of units (drivers) in the formation.	
formationPosition person Return position of unit in the formation	
formationTask person Return the current task of the unit in the formation.	
formatText [format, arg1, arg2, ...] Creates a structured text by replacing %1, %2, etc. in format with plain or structured texts given as arguments.	txt = formatText ["Image: %1", image "data\isniper.paa"]
formLeader unit Returns the formation leader of a given unit. This is often the same as the group leader, but not always, for example in cases when a unit is	? (formLeader player != leader player) : hint "The formation leader is different to the group leader!" Ergebnis X

ordered to follow another unit.	
friendly Friendly side (friendly to all units).	
for "_var" from b Continue sequence of for var command.	for "_x" from 10 to 20 do {..code..}
fuel vehicle Checks how much fuel is left in the gas tank of a vehicle.	?(fuel (vehicle player) == 0) : hint "The vehicle is out of fuel!"
getArray config Extract array from config entry.	_array = getArray (configFile >> "CfgVehicles" >> "Thing" >> "threat")
getDammage obj Returns the object damage in the range from 0 to 1.	getDammage player
getDir obj Returns the object heading in the range from 0 to 360.	getDir player
object getHideFrom enemy Returns the hiding position in format Position. If enemy is null it is some position in front of the object or enemy position.	
getMarkerColor marker Returns the color of a given map marker.	"MarkerOne" setMarkerColor "ColorBlack" _color = getMarkerColor "MarkerOne" Ergebnis "ColorBlack"
getMarkerPos marker Returns the position of a given marker. [x,y,z] Argument 3 (height above ground) is always zero. If a non-existing marker is referenced the values returned are [0,0,0].	"MarkerOne" setMarkerPos [200,100] _pos = getMarkerPos "MarkerOne"
getMarkerSize marker Returns the size of a given marker.	"MarkerOne" setMarkerSize [100,200] _size = getMarkerSize "MarkerOne" Ergebnis [100,200]
getMarkerType marker Returns the type of a given marker.	"MarkerOne" setMarkerType "Destroy" _type = getMarkerType "MarkerOne" Ergebnis "Destroy"
getNumber config Extract number from config entry.	_value = getNumber (configFile >> "CfgVehicles" >> "Thing" >> "maxSpeed")
getPos obj Returns the object position in format Position.	getPos player
getPosASL obj Returns the object position in format PositionASL.	getPosASL player
object getSpeed speedMode Get the speed for the given speed mode. SpeedMode can be: "AUTO" "SLOW"	

"NORMAL" "FAST"	
getText config Extract text from config entry.	_array = getText (configFile >> "CfgVehicles" >> "Thing" >> "icon")
object getVariable name Return the value of variable in the variable space of given object.	myTruck getVariable "myVariable";
getWPPos [group, number] Returns the position of a selected waypoint of a given group. Waypoints include only those which were placed in the mission editor.	[group1,1] setWPPos [200,600,0] _pos = getWPPos [group1,1] Ergebnis [200,600,0]
unit(s) glanceAt position Control what the unit is glancing at (target or position) (format Position). How frequently the unit is glancing there depends on behaviour.	someSoldier glanceAt otherSoldier or otherSoldier glanceAt markerPos "markerOne"
unit globalChat text Make a unit send a text message over the global radio channel.	soldierOne globalChat "Show this text"
unit globalRadio message Make a unit send a message over the global radio channel. The message is defined in the description.ext of the mission and may contain text and sound.	soldierOne globalRadio "messageOne"
goto label In SQS scripts only: Go to given label. String argument is used here. Be sure to use double quotes around label name in goto. Define the label with #. Note that Labels are not case sensitive and that labels are searched for from the top of the script, so multiple occurrences of a label will only result in the top most one ever being found.	
group unit Returns the group a unit is assigned to.	playerGrp = group player
unit groupChat text Make a unit send a text message over the group radio channel.	soldierOne groupChat "Show this text"
unit groupRadio message Make a unit send a message over the group radio channel. The message is defined in the description.ext of the mission and may contain text and sound.	soldierOne groupRadio "messageOne"
grpNull A non-existing group. This value is not equal to anything, including itself.	group player == grpNull Ergebnis false
gunner vehicle Returns the gunner of a vehicle.	(gunner tank) action ["getout",tank]
halt Stops the program into a debugger. Example halt	halt
handsHit soldier Checks if a soldier's hands are hit, which results in inaccurate aiming.	? (handshit player == 1) : player globalChat "Ouch! Don't shoot at my hands dammit!"

unit hasWeapon weapon Checks if a unit has the given weapon.	?!(player hasWeapon "M16") : player addWeapon "M16"
object hideBehindScripted scriptedHideBehind When set to true it disables the default engine hiding behavior.	unit hideBehindScripted true
hideBody person Hides the body of the given person.	hideBody player
hierarchyObjectsCount The number of objects in hierarchy.	
hint text Outputs a hint message on the left upper corner of the screen together with a clinging sound. The text may contain several lines.	hint "Press W to move forward. \nPress S to move backwards. \n\nUse the mouse to turn right or left." Ergebnis Press W to move forward. Press S to move backwards. Use the mouse to turn right or left.
hintC text Works the same way as hint, except the text is displayed in the middle of the screen. The game is paused until the player presses "Continue".	hintC "Press W to move forward"
hintCadet text Shows a text hint only when using cadet mode. The text can contain several lines. \n is used to indicate the end of a line.	hintCadet "Press W to move forward"
title hintC [text1, text2, ...] Creates a hint dialog with the given title and text. Texts can be plain or structured. Note: Not working properly.	
title hintC text Creates a hint dialog with the given title and text. Note: Not working properly.	
title hintC text Creates a hint dialog with the given title and text. Note: Not working properly.	
control htmlLoad filename Load HTML from file to given control.	_control htmlLoad "briefing.html"
if condition First part of if command.	if (a>b) then {a=b}
image filename Creates a structured text containing the given image.	txt1 = image "data\isniper.paa"
soldier in vehicle Checks whether the soldier is mounted in the vehicle.	player in jeepOne
x in Array Checks whether x is equal to any element in the array.	1 in [0, 1, 2] Ergebnis true
fireplace inflame true Control fireplace burning. Set inflame to true (on) or false (off).	fireplaceOne inflame true
inflamed fireplace	_it = inflamed fireplaceOne

Check if fireplace is inflamed (burning) or not.	
inGameUISetEventHandler [handler name, function] Sets given event handler of in-game UI.	
inheritsFrom config Returns base entry of config entry.	_base = inheritsFrom (configFile >> "CfgVehicles" >> "Car")
initAmbientLife Initialize the ambient life.	
[object, lod name] intersect [begin, end] Find named selection in object which is in specified lod intersected by given section of a line.	[tank, "VIEW"] intersect [[1500, 1500, 2], [1550, 1500, 2]]
isArray config Check if config entry represents array.	_ok = isArray (configFile >> "CfgVehicles")
isClass config Check if config entry represents config class.	_ok = isClass (configFile >> "CfgVehicles")
isEngineOn vehicle Returns true if engine is on, false if it is off.	_it = isEngineOn carOne
isFormationLeader person Returns true if the specified person is subgroup leader.	
isHidden person Return whether the person is hidden (reached the hiding position).	
isHideBehindScripted vehicle Return whether the vehicle has set the hideBehindScripted true.	
isKeyActive keyName Checks whether the given key is active in the current user profile. See keys, keysLimit and doneKeys in the description.ext file of the missions.	_ok = isKeyActive "M04"
object isKindOf typeName Checks whether the object is of the given type.	vehicle player isKindOf "Tank"
isMarkedForCollection object Checks whether the object is marked for weapons collection.	marked = isMarkedForCollection truck
isnil variable Tests whether the variable is null. The function returns true if the variable is null and false if it's not.	if (isnil ("_pokus")) then {_pokus=0;}
isNull control Checks whether the value is equal to controlNull. Note: a==controlNull does not work, because controlNull is not equal to anything, even to itself.	isNull controlNull
isNull display Checks whether the value is equal to displayNull. Note: a==displayNull does not work, because displayNull is not equal to anything, even to itself.	isNull displayNull

isNull grp Checks whether the value is equal to grpNull. Note: a == grpNull does not work, because grpNull is not equal to anything, not even to itself.	isNull group player
isNull obj Checks whether the value is equal to objNull. Note: a==objNull does not work, because objNull is not equal to anything, even to itself.	isNull objNull
isNumber config Check if config entry represents number.	_ok = isNumber (configFile >> "CfgVehicles")
isPlayer person Check if given person is a human player.	
isText config Check if config entry represents text.	_ok = isText (configFile >> "CfgVehicles")
unitArray join group Join all units in the array to given group. Notes: Total number of group members must not exceed 12 (in OFP!). This function is unsupported in MP in version 1.33 and before, now it has to be executed where the given unit is local.	[unitOne, unitTwo] join player
person kbAddDatabase filename Register knowledge base database to given person.	_unit kbAddDatabase "chat.txt"
person kbAddDatabaseTargets filename Register target list knowledge base database to given person.	_unit kbAddDatabase "chat.txt"
person kbAddTopic [name, filename(, task type)] Register conversation topic to given person.	
person kbHasTopic name Check if conversation topic was registered to given person.	
person kbRemoveTopic topicname Unregister conversation topic from given person.	
person kbTell [receiver, topic, sentence id, [argument name, argument value, argument text, argument speech], ...] Make the person tell to the receiver the sentence.	
keyImage dikcode Returns a structured text, containing an image or name (if no image is found) of the button, on the keyboard, mouse or joystick, with the given code.	name = keyImage 28 result is "Enter"
keyName dikCode Returns the name of a button (on the keyboard, mouse or joystick) with the given code.	name = keyName 28 result is "Enter"
unit knowsAbout target Check if (and by how much) unit knows about target. If unit is vehicle, vehicle commander is considered.	_kv = soldierOne knowsAbout jeepOne
helicopter land mode Force helicopter landing. Landing mode may be:	cobraOne land "LAND"

"LAND" (complete stop) "GET IN" (hovering very low, for another unit to get in) "GET OUT" (hovering low, for another unit to get out)	
lbAdd [idc, text] Adds an item with the given text to the listbox or combobox with id idc of the topmost user dialog. It returns the index of the newly added item.	_index = lbAdd [101, "First item"]
control lbAdd text Adds an item with the given text to the given listbox or combobox. It returns the index of the newly added item.	_index = _control lbAdd "First item"
lbClear idc Clear all items in listbox or combobox with id idc of topmost user dialog.	lbClear 101
lbClear control Clears all items in the given listbox or combobox.	lbClear _control
lbColor [idc, index] Returns the text color of the item with the given index of the listbox or combobox with id idc of the topmost user dialog. The color is returned in format Color.	_colour = lbColor [101, 0]
control lbColor index Returns the text color of the item with the given index of the given listbox or combobox. Colour is in format Color.	_color = _control lbColor 0
lbCurSel idc Returns the index of the selected item of the listbox or combobox with id idc of the topmost user dialog.	_index = lbCurSel 101
lbCurSel control Returns the index of the selected item of the given listbox or combobox.	_index = lbCurSel _control
lbData [idc, index] Returns the additional text (invisible) in an item with the given index of the listbox or combobox with id idc of the topmost user dialog.	_data = lbData [101, 0]
control lbData index Returns the additional text (invisible) in an item with the given index of the given listbox or combobox.	_data = _control lbData 0
lbDelete [idc, index] Removes the item with the given index from the listbox or combobox with id idc of the topmost user dialog.	lbDelete [101, 0]
control lbDelete index Removes the item with the given index from the given listbox or combobox.	_control lbDelete 0
control lbIsSelected index Check whether given row of the given listbox is	_selected = _control lbIsSelected 0

selected.	
lbPicture [idc, index] Returns the picture name of the item with the given index of the listbox or combobox with id idc of the topmost user dialog.	_picture = lbPicture [101, 0]
control lbSetPicture index Returns the picture name of the item with the given index of the given listbox or combobox.	_picture = _control lbPicture 0
lbSelection control Returns the array of selected rows indices in the given listbox	_indices = lbSelection _control
lbSetColor [idc, index, color] Sets the color of the item with the given index of the listbox or combobox with id idc of the topmost user dialog to color. Colour is in format Color.	lbSetColor [101, 0, [0, 1, 0, 0.5]]
control lbSetColor [index, color] Sets the text color of the item with the given index of the given listbox or combobox. Colour is in format Color.	_control lbSetColor [0, [0, 1, 0, 0.5]]
lbSetCurSel [idc, index] Selects the item with the given index of the listbox or combobox with id idc of the topmost user dialog.	lbSetCurSel [101, 0]
control lbSetCurSel index Selects the item with the given index of the given listbox or combobox.	_control lbSetCurSel 0
lbSetData [idc, index, data] Sets the additional text (invisible) in the item with the given index of the listbox or combobox with id idc of the topmost user dialog to the given data.	lbSetData [101, 1, "#1"]
control lbSetData [index, data] Sets the additional text (invisible) in the item with the given index of the given listbox or combobox to the given data.	_control lbSetData [1, "#1"]
lbSetPicture [idc, index, name] Sets the picture in the item with the given index of the listbox or combobox with id idc of the topmost user dialog. Name is the picture name. The picture is searched for in the mission directory, the dtaExt subdirectory of the campaign directory and the dtaExt directory and the data bank (or directory).	lbSetPicture [101, 0, "iskoda"]
control lbSetPicture [index, name] Sets the picture in the item with the given index of the given listbox or combobox. Name is the picture name. The picture is searched for in the mission directory, the dtaExt subdirectory of the campaign directory and the dtaExt directory and the data bank (or directory).	_control lbSetPicture [0, "iskoda"]

control lbSetSelected [index, selected] Set the selection state of the given row of the given listbox. Listbox must support multiple selection.	_control lbSetSelection [0, true]
lbSetValue [idc, index, value] Sets the additional integer value in the item with the given index of the listbox or combobox with id idc of the topmost user dialog to the given value.	lbSetValue [101, 0, 1]
control lbSetValue [index, value] Sets the additional integer value in the item with the given index of the given listbox or combobox to the given value.	_control lbSetValue [0, 1]
lbSize idc Return number of items of listbox or combobox with id idc of topmost user dialog.	_n = lbSize 101
lbSize control Returns the number of items in the given listbox or combobox.	_n = lbSize _control
lbText [idc, index] Returns the shown text in the item with the given index of the listbox or combobox with id idc of the topmost user dialog.	_text = lbText [101, 0]
control lbText index Returns the shown text in the item with the given index of the given listbox or combobox.	_text = _control lbText 0
lbValue [idc, index] Returns the additional integer value in the item with the given index of the listbox or combobox with id idc of the topmost user dialog.	_value = lbValue [101, 0]
control lbValue index Returns the additional integer value in the item with the given index of the given listbox or combobox.	_value = _control lbValue 0
leader unit Returns the group leader for the given unit or group. For dead units, objNull is returned.	leader group player == leader player
group leaveVehicle vehicle Ceases the using of the vehicle in the group. It unassigns all grouped units from the vehicle.	groupOne leaveVehicle jeepOne
unit leaveVehicle vehicle Ceases the using of the vehicle in the group. It unassigns all grouped units from the vehicle.	soldierOne leaveVehicle jeepOne
light lightAttachObject [object, position] Attach light to given object (at given position).	
lightDetachObject light Detach light from object.	
lightIsOn lamppost Check if lamppost is on (shining). Possible values are: "ON" "OFF" "AUTO" (auto is only on during the night).	lightIsOn nearestObject [player, "StreetLamp"] != "OFF" _it = lightIsOn object 159582

object limitSpeed speed Limit speed of given vehicle to given value (in km/h).	
lineBreak Creates a structured text containing a line break.	txt3 = lineBreak
list trigger List of units that would activate given trigger. For trigger of type "Not present" the list is the same as that returned for "present".	_tlist = list triggerOne
ln x Natural logarithm of x.	_nlog = ln 10 Ergebnis 2.302
loadFile filename Return content of given filename.	loadFile "myFunction.sqf"
person loadIdentity name Loads person's identity from Objects.sav file in campaign directory (from entry name).	player loadIdentity "playerIdentity"
object loadStatus name Loads object's properties from Objects.sav file in campaign directory (from entry name).	player loadStatus "playerState"
local obj Check if given unit is local on the computer in Multiplayer games (see Locality in Multiplayer for general concepts). This can be used when some activation fields or scripts need to be performed only on one computer. In Singleplayer all objects are local. Note: All static objects are local everywhere.	local unitName
localize stringName Replace string with given stringName with corresponding localized text from Stringtable.csv.	localize "STR_DN_FROG"
vehicle lock lock Lock vehicle (disable mounting / dismounting) for player.	jeepOne lock true
locked unit Check if vehicle is locked for player. If it is locked, player cannot mount / dismount without order.	_it = locked jeepOne
group lockWP lockWP Disable switching to next waypoint (current waypoint will never complete while lockwp is used). Sometimes used during cut-scenes.	groupOne lockWP true
log x Base-10 logarithm of x.	_log = log 10Result is 1
unit(s) lookAt position Control what the unit is looking at (target or position).	someSoldier lookAt otherSoldier or otherSoldier lookAt markerPos "markerOne"
magazines vehicle Returns array of type names of all vehicle's magazines.	_mags = magazines player

mapAnimAdd frame Add next frame to map animation. Format of frame is [time, zoom, position], format of position is Position2D.	mapAnimAdd [1, 0.1, markerPos "anim1"]
mapAnimClear Clear map animation.	
mapAnimCommit Play map animation.	
mapAnimDone Check if map animation is finished.	
markerColor markername Get marker colour. See setMarkerColor. Note: This function is identical to getMarkerColor.	? markerColor "MarkerOne" == "ColorRed" : player setFace "Marilyn"
markerDir markerName Get marker direction.	_mPos = markerDir "markerOne"
markerPos markerName Get marker position [x,z,y]. Note: This function is identical to getMarkerPos.	_mPos = markerPos "markerOne"
markerSize markerName Get marker size. Note: This function is identical to getMarkerSize.	_mSize = markerSize "MarkerOne"
markerText markerName Get marker text.	_mDesc = markerText "markerOne"
markerType markerName Get type of marker. Note: This function is identical to getMarkerType.	? markerType "MarkerOne" == "Dot" : "MarkerOne" setMarkerType "Arrow"
Max The greater of a,b	3 max 2 Ergebnis ist 3
Min The smaller of a,b	3 min 2 Ergebnis ist 2
missionConfigFile Return root of mission description.ext entries hierarchy.	
missionName Return name of current mission. Works only in multiplayer, in singleplayer returns empty String.	
missionStart Return time when mission started in format [year, month, day, hour, minute, second]. Works only in multiplayer, in singleplayer all are values equal to zero → [0,0,0,0,0,0]	
a mod b Remainder of a divided by b.	_rem = 3 mod 2 Ergebnis ist 1
object modelToWorld modelPos Converts position from object model space to world space.	
group move pos Creates a move waypoint on given position (format Position or Position2D) and makes it an actual group waypoint.	groupOne move position player
soldier moveInCargo vehicle	soldierOne moveInCargo jeepOne

Move soldier into vehicle cargo position (Immediate, no animation).	
soldier moveInCargo [vehicle, CargoIndex] Moves the soldier into a vehicle's specified cargo position. (Immediately, without animation).	soldierOne moveInCargo [jeepOne, 1]
soldier moveInCommander vehicle Move soldier into vehicle commander position (Immediate, no animation).	soldierOne moveInCommander tankOne
soldier moveInDriver vehicle Move soldier into vehicle driver position (Immediate, no animation).	soldierOne moveInDriver tankOne
soldier moveInGunner vehicle Move soldier into vehicle gunner position (Immediate, no animation).	soldierOne moveInGunner tankOne
soldier moveInTurret [vehicle, turret path] Moves the soldier into the vehicle's turret. (Immediately, without animation).	soldierOne moveInTurret [tank, [0, 0]]
obj moveTarget posDescription Change information about a target.	target moveTarget [position player,1,1]
person moveTo position Low level command to person to move to given position.	
moveToCompleted person Check if latest low level moveTo command is finished.	
moveToFailed person Check if latest low level moveTo command failed.	
musicVolume Checks the current music volume (set by fadeMusic)	
name object The name given to a unit using the setIdentity instruction or selected randomly by the game engine if setIdentity has not been used on the unit. If used on vehicle, name of first crew member (in order commander, driver, gunner). If used on vehicle, name of first crew member (in order commander, driver, gunner).	_name = name vehicle player
nearestBuilding obj Nearest building to given object.	_nBuilding = nearestBuilding player
nearestObject pos Nearest object of given type to given position. For nearestObject pos, pos may be [x, y ,z, "type"] or [object, "type"] or even [x, y, z]. Before ArMA the funcion matched only objects with exactly the type given. Since ArMA, any object derived from the type is found as well.	_nObject = nearestObject [player, "Tank"] _nObject = nearestObject [player, "StreetLamp"]
pos nearestObject id Find object nearest to given position with given	_nObject =position player nearestObject 123456

<p>Visitor id. Position should be accurate, function is not guaranteed to find objects further than 50 m away.</p>	
<p>position nearestObject type Find object nearest to given position with given type.</p>	<pre>_nObject = position player nearestObject "StreetLamp"</pre>
<p>nearestObjects [unit, class, range] Returns a list of nearest objects of the given types to the given position or object, within the specified distance. Pos may be using format [x,y,z, ["type",...], limit] or [object, ["type",...], limit]. If more than one object is found they will be ordered according to their distance (i.e. the closest one will be first in the array).</p>	<pre>nearestObjects [player, ["house"], 200] nearestObjects [player, ["Car", "Tank"], 200]</pre>
<p>position nearObjects radius Find objects in the circle with given radius. If typeName is given, only objects of given type (or its subtype) are listed.</p>	<pre>_list= [_xpos,_ypos] nearObjects ["House",20] _list = position player nearObjects 50</pre>
<p>needReload vehicle Return how much vehicle wants to reload its weapons.</p>	
<p>nextWeatherChange Return the time (in seconds) when the next weather change will occur.</p>	
<p>nil Nil value. This value can be used to undefine existing variables.</p>	<pre>variableToDestroy = nil</pre>
<p>not a not a. not a is exactly the same as ! a</p>	<pre>not false Ergebnis true</pre>
<p>objNull Non-existent object. This value is not equal to anything, not even to itself.</p>	<pre>_it = player == objNull</pre>
<p>objective objStatus status Sets the status of an objective that was defined in briefing.html . Status may be one of: "ACTIVE" "FAILED" "DONE" "HIDDEN" To refer to an objective that is named "OBJ_1", for example, use only the index number in this command (i.e. "1" objStatus "HIDDEN").</p>	<pre>"1" objStatus "DONE" (Marks the objective named "OBJ_1" as completed.)</pre>
<p>onBriefingGear sound Define sound (voice) played the first time when section Gear in briefing is selected.</p>	<pre>onBriefingGear "GearVoiceOver"</pre>
<p>onBriefingGroup sound Define sound (voice) played the first time when section Group in briefing is selected.</p>	<pre>onBriefingGroup "GroupVoiceOver"</pre>
<p>onBriefingNotes sound Define sound (voice) played the first time when section Notes in briefing is selected.</p>	<pre>onBriefingNotes "NotesVoiceOver"</pre>

onBriefingPlan sound Define sound (voice) played the first time when section Plan in briefing is selected.	onBriefingPlan "PlanVoiceOver"
onBriefingTeamSwitch sound Defines a sound (voice) that is played the first time when the Team switch section in the briefing is selected.	onBriefingTeamSwitch "TeamSwitchVoiceOver"
onMapSingleClick command Define action performed when user clicks in map. Command receives: _pos Array position _units Array selected units _shift, _alt Boolean key state If click is processed, command should return true.	onMapSingleClick "" "SoldierWB" createUnit [_pos, group player]" Creates a soldier unit at the position clicked. or onMapSingleClick "grp1 move _pos; onMapSingleClick {}" Orders "grp1" to move to position clicked. Disables further map-click actions.
onPlayerConnected statement This statement is launched whenever a player is connected to a MP session. Variables _id and _name are set.	
onPlayerDisconnected statement This statement is launched whenever a player is disconnected from a MP session. Variables _id and _name are set.	
a or b	if (OBJ1) or (enemycount==0) then {hint "you win" }
unitArray orderGetIn order Force all units in the array to get in or out of their assigned vehicles. Units must be assigned to a vehicle before this command will do anything.	[unitOne, unitTwo] orderGetIn true or [unitOne, unitTwo] orderGetIn false
overcast Return the current overcast.	setOvercast (overcast + 0.1)
overcastForecast Return the overcast forecast.	
parseNumber text Parse string containing real number.	number = parseNumber "0.125"
parseText text Creates a structured text by parsing the given XML description.	txt = parseText "First line Secon d line"
pi pi (180 degrees converted to radians).	_a = 2*pi Ergebnis 6.2830
pickWeaponPool object Transfer weapons and magazines from cargo of object into weapon pool (used in campaign to transfer weapons into next mission).	
playerRespawnTime Return the player remaining time to respawn.	@playerRespawnTime == 0
playerSide Returns the player's side. This is valid even when the player controlled person is dead (a difference	

from side player).	
playersNumber side Return count of players playing on given side. Works only in multiplayer, in singleplayer always returns 0.	
playMission [campaign, mission] or [campaign, mission, skipBriefing] The mission is launched (from the main menu). Both campaign and mission are given as their directory name. If the campaign is empty, a single mission is launched. If skipBriefing is true, the intro and briefing are skipped.	playMission ["XOutrage", "x05Negotiator.Noce"]
soldier playMove moveName When used on person, smooth transition to given move will be done.	soldierOne playMove "Stand"
playMusic name Plays music defined in the description.ext file.	playMusic "musicname"
playMusic nameAndpos Plays music defined in the description.ext file. The format of nameAndPos is [name,position]. Position is in seconds.	playMusic ["Track13", 30]
playSound name Play sound defined in Description.ext.	playSound "soundname"
position object Synonym for getPos. Object position in format Position.	pPos = position player
map posScreenToWorld [x,y] Convert screen coordinates in map to world coordinates.	
positionCameraToWorld position Transform position from camera coordinate space to world coordinate space.	_worldPos = positionCameraToWorld _cameraPos
map posWorldToScreen position Convert world coordinates to screen coordinates in map.	map posWorldToScreen position
precision entity Return the precision of the given entity.	
preloadCamera position Preload all textures and models around given position (format Position) to avoid visual artifacts after camera is moved. Should be used before any abrupt camera change / cut. Returns true once all data are ready.	spawn { waitUntil preloadCamera markerPos "cam_location_2" }
distance preloadObject object Preload all textures, materials and proxies needed to render given object. Object can be determined either by config class name, or by object id. Returns true once all data is loaded and ready.	spawn { waitUntil 10 preloadObject "SoldierW" } or spawn { waitUntil 10 preloadObject leader player }
preloadSound sound Makes sure that a sound can start playing without any delay once it is needed.	

Note:Not implemented yet - currently does nothing.	
preloadTitleObj effect Object title - argument uses format ["text","type",speed] or ["name","type"]. Speed is ignored. Preload data the object can be defined in the Description.ext file.	preloadTitleObj ["BISLogo","plain"]
preloadTitleRsc effect Resource title - argument uses format ["name","type",speed] or ["name","type"]. Speed is ignored. Preload data the resource can be defined in the Description.ext file.	preloadTitleRsc ["BIS", "PLAIN"]
preprocessFile filename Returns preprocessed content of given file. Preprocessor is C-like, supports comments using // or /* and */ and macros defined with #define.	preprocessFile "myFunction.sqf"
preprocessFileLineNumbers filename Returns the preprocessed content of the given file. The preprocessor is C-like, it supports comments using // or /* and */ and macros defined with #define.	preprocessFileLineNumbers "myFunction.sqf" Ergebnis Result is "if a>b then {a} else {b}"
primaryWeapon vehicle Returns name of vehicle's primary weapon (empty string if none).	pWeap = primaryWeapon player
private variable Introduces one or more local variables in the innermost scope.	private ["_varname"]; private ["_varname1", "_varname2"];
private variable Introduces one or more local variables in the innermost scope.	
processInitCommands Process statements stored using setVehicleInit. The statements will only be executed once even if processInitCommands is called multiple times.	
publicVariable varName Broadcast variable value to all computers. Only type Number is supported in version 1.33 and before. Following Types are supported since version 1.34: Number Boolean Object Group Following Types are supported since version Arma v1.00: String	publicVariable "CTFscoreOne"
putWeaponPool obj Transfer weapons and magazines from weapon pool into cargo of object obj. Used in campaign to transfer weapons into next mission.	
queryMagazinePool name Return number of magazines of type name in magazine pool (used in campaign to transfer magazines into next mission).	
rad x	_radians = rad 180

Convert x from degrees to radians. 360 degrees is equal to 2 multiplied with pi.	Ergebnis 3.1415
radioVolume Checks the current radio volume (set by fadeRadio).	
rain Return the current rain.	
random x Random real value from 0 (inclusive) to x (not inclusive).	_rNumber = random 1
rank unit Return the rank of the given unit.	
rating unit Check unit rating. Rating is increased for killing enemies, decreased for killing friendlies, can be changed by mission designer.	_score = rating player
reload vehicle Reload all weapons	
reloadEnabled vehicle Check whether magazine is reloaded whenever emptied.	
unit removeAction index Remove action with given id index.	player removeAction 0
object removeAllEventHandlers handlerType Removes all event handlers of given type that were added by addEventHandler.	[[player]] removeAllEventHandlers "killed"
removeAllWeapons unit Remove all weapons and magazines of the unit. On vehicles only ammo is removed	removeAllWeapons player
object removeEventHandler handler Removes event handler added by addEventHandler. Format of handler is [type, index]. Index is returned by addEventHandler. When any handler is removed, all handler indices higher than the deleted one should be decremented.	player removeEventHandler ["killed", 0]
unit removeMagazine magazineName Remove magazine from the unit. Note: You may create invalid combinations with this function. When doing so, application behaviour is undefined.	player removeMagazine "M16"
unit removeMagazines magazineName Remove all magazines of given type from the unit. Note: You may create invalid combinations with this function. When doing so, application behaviour is undefined.	player removeMagazines "M16"
removeSwitchableUnit person Remove a unit from the list of units available for Team Switch.	
unit removeWeapon weaponName	player removeWeapon "M16"

Remove weapon from the unit. Note: You may create invalid' combinations with this command. When doing so, application behaviour is undefined.	
requiredVersion version Check if version of application is available. If not, show warning message and return false. Version of format Major.Minor, e.g. 1.30	requiredVersion "1.30"
resistance Resistance side. When used in a format statement (hint format["%1",resistance]), the string returned is "GUER".	
array resize count Change array size. Can be used to add or remove elements from an array.	arrayOne resize 2
Nothing Set vehicle as respawnable in MP games. Delay is respawn delay, default respawnDelay from Description.ext is used. Count tells how many respawns is processed (default unlimited).	car respawnVehicle [5.0, 3]
group reveal unit Reveals a unit to a group or another unit. The knowledge value will be set to the highest level any unit of the revealing side has about the revealed unit. If the revealing side has no knowledge about the revealed unit, the value will be set to 1.	soldierOne reveal soldierTwo (soldierOne received information about soldierTwo)
round x Rounds up or down to the closest integer of x.	round 5.25, result is 5 or round 5.55, result is 6
runInitScript Launch init.sqs script.	
saveGame Autosave game (used for Retry).	
person saveIdentity name Saves person's identity to Objects.sav file in campaign directory as entry name.	player saveIdentity "playerid"
object saveStatus name Saves object's properties to Objects.sav file in campaign directory as entry name.	player saveStatus "playerstate"
saveVar varname Save variable value into the campaign space. This variable is available to all following missions in the campaign	saveVar "varOne"
unit say speechName Format of speechName is [sound, maxTitlesDistance] or [sound, maxTitlesDistance, speed]. Unit will say given sound.	(units player select 1) say ["Sound",5]

<p>When the unit is a person, it will also perform corresponding lipsync effect provided an appropriate .lip file has been created for this sound.</p> <p>A unit that has been killed or does not exist will not say anything.</p> <p>Compare this with playSound which will always play a sound at the location of the player.</p> <p>If the camera is not within given range, title is not shown and the sound will not be heard.</p> <p>Sound is defined in Description.ext.</p>	
<p>unit say speechName</p> <p>The unit will play the given sound.</p> <p>If the unit is a person, it will also perform the corresponding lipsync effect.</p> <p>Sound is defined in Description.ext. .</p>	soldierOne say "speechId"
<p>scopeName name</p> <p>Defines name of current scope. Name is visible in debugger, and name is also used as reference in some commands. Scope name can be defined only once per scope.</p>	
<p>score unit</p> <p>MP: Gibt die Punkte der Einheit zurück.</p>	pScore = score player
<p>scriptDone script</p> <p>Prüft ob das skript bereits ausgeführt/beendet wurde</p>	
<p>scudState scudname</p> <p>Status der angegeben Scud. Status kann sein:</p> <p>0 - Nicht Aktiv</p> <p>1 - Start Vorbereitungen</p> <p>2 - Start vorbereitet</p> <p>3 - Zündung</p> <p>4 - Gestartet</p>	? scudState scudName >= 3 : hint "RUN!!!!"
<p>secondaryWeapon vehicle</p> <p>Gibt den Namen der Sekundärwaffe eines Fahrzeugs zurück. (Leeren String wenn keine vorhanden)</p>	sWeap = secondaryWeapon player
<p>array select index</p> <p>Selects index element of the array.</p> <p>Index 0 denotes the first element, 1 the second, etc. If index has decimal places <= x.5 it gets floored, otherwise ceiled.</p>	<p>[1,2,3,4] select 2</p> <p>- result is 3</p> <p>or</p> <p>[1,2,3,4] select true</p> <p>- result is 2.</p> <p>Alternative method. If the index is false, this selects the first element of the array. If it is true, it selects the second one. The reason for this is that false is an alias for 0 and true an alias for 1. Obsolete syntax.</p> <p>or</p> <p>position player select 2</p>

	- result is Z coordinate of player position (see Position for more details)
config select index Returns subentry with given index.	(configFile >> "CfgVehicles") select 0
object selectionPosition selection name Search for selection in the object model (first in the memory level, then in other levels). Returns position in model space.	
group selectLeader unit Select group leader.	group player selectLeader player
selectPlayer unit Switch player to given unit.	selectPlayer aP
unit selectWeapon weapon Select given weapon. For weapon values see cfgWeapons.	soldier1 selectWeapon "WeaponName"
object sendSimpleCommand command Sends a simple command to the vehicle's driver / gunner.	vehicle player sendSimpleCommand "STOP"
array set element Format of element is [index, value]. Changes an element of given array. If element does not exist, resize index+1 is called to create it.	arrayOne set [0, "Hello"]
setAccTime accFactor Set time acceleration coefficient. May be also used to slow time in cutscenes. This command does NOT work in multiplayer.	setAccTime 0.1
vehicle setAmmoCargo amount Set amount or ammo resources in cargo space of rearm vehicle. Ammo resource is used to resupply vehicles that take ammo. Soldiers use individual magazines instead. Amount: 1 is full cargo.	
setAperture set Sets custom camera aperture (-1 to do it automatically).	
text setAttributes [name1, value1, name2, value2, ...] Returns a structured text created by the given structured or plain text by setting attributes to the given values.	txt = img setAttributes ["image", "data\iSoldier.paa"]
group setBehaviour behaviour Set group behaviour mode. Behaviour is one of: "CARELESS" "SAFE" "AWARE" "COMBAT" "STEALTH".	group1 setBehaviour "safe"
trigger setCameraEffect [name, position] Name defines the effect type (a subclass of	trigger setCameraEffect ["ZoomIn", "FRONT"]

<p>CfgCameraEffects.Array):</p> <p>"TERMINATE","INTERNAL","ZOOMIN","ZOOMINSLOW","AROUND","AROUNDSLOW","ZOOMANDAROUND","AROUNDANDZOOMINSLOW","ZOOMINS","FIXED","FIXEDWITHZOOM","EXTERNAL"</p> <p>Or "\$TERMINATE\$" to cancel the current effect.</p> <p>Position is camera placement: "TOP", "LEFT", "RIGHT", "FRONT", "BACK", "LEFT FRONT", "RIGHT FRONT", "LEFT BACK", "RIGHT BACK", "LEFT TOP", "RIGHT TOP", "FRONT TOP", "BACK TOP" or "BOTTOM".</p> <p>Alternativer Syntax: waypoint setCameraEffect [name, position]</p>	
<p>entity setCameraInterest interest Set camera interest for given entity.</p>	<p>_soldier setCameraInterest 50</p>
<p>person setCaptive captive Mark unit as captive. If unit is a vehicle, commander is marked. A captive is neutral to everyone, and will not trigger "detected by" conditions for its original side.</p>	<p>soldier1 setCaptive true</p>
<p>group setCombatMode mode Set group combat mode (engagement rules). Mode may be one of:</p> <p>"BLUE" (Never fire) "GREEN" (Hold fire - defend only) "WHITE" (Hold fire, engage at will) "YELLOW" (Fire at will) "RED" (Fire at will, engage at will)</p>	<p>group1 setCombatMode "BLUE"</p>
<p>object setDamage damage Damage / repair object. Damages or repairs the object. Damage 0 means fully functional, damage 1 means completely destroyed / dead.</p>	<p>soldier1 setDamage 1</p>
<p>object setDammage damage Damage / repair object. Damages or repairs the object. Damage 0 means fully functional, damage 1 means completely destroyed / dead.</p>	<p>soldier1 setDammage 1</p>
<p>setDate [year, month, day, hour, minute] Sets the actual mission date and time.</p>	<p>setDate [1986, 2, 25, 16, 0]</p>
<p>object setDestination [position, planningMode, forceReplan] Set the destination for path planning of the pilot.</p>	
<p>object setDir heading</p>	<p>soldier1 setDir 45</p>

<p>Sets object heading. Angles are measured in degrees clockwise from north. The accepted heading range is from 0 to 360 Negative angles represent a counter-clockwise angle and the angle can be of any size.</p>	<p>Will set soldier1 to face North East or soldier1 setDir -675 Will also set soldier1 to face North East</p>
<p>particleSource setDropInterval interval Set interval of emitting particles from particle source.</p>	<p>_source setDropInterval 0.05</p>
<p>trigger setEffectCondition statement The statement is executed when the trigger or waypoint is activated and the effects are launched depending on the result.</p> <p>If the result is a boolean and true, the effect is launched. If the result is an object, the effect is launched if the result is the player or the player vehicle. If the result is an array, the effect is launched if the result contains the player or the player vehicle.</p> <p>Alternativer Syntax: waypoint setEffectCondition statement</p>	<p>trigger setEffectCondition "thisList"</p>
<p>person setFace face Set person's face.</p>	<p>soldier1 setFace "face10"</p>
<p>person setFaceAnimation blink Set facial animation phase (eye blinking), blink is in the range from 0 to 1</p>	<p>soldier1 setFaceAnimation 0.5</p>
<p>flag setFlagOwner owner Set flag owner. When owner is set to objNull or any object other than a unit of class man or logic, flag is returned to the flagpole. A flag owned by a logic has no visual representation.</p>	<p>flag1 setFlagOwner soldier1</p>
<p>flag setFlagSide side Set flag side.</p>	<p>flag1 setFlagSide east</p>
<p>flag setFlagTexture texture Set flag texture. If texture is "", flag is not drawn.</p>	<p>flagE setFlagTexture "\ca\misc\data\sever_vlajka.paa" or flagW setFlagTexture "\ca\misc\data\usa_vlajka.paa"</p>
<p>time setFog fog Changes the fog smoothly during the given time (in seconds). A time of zero means there will be an immediate change. A fog level of zero is minimum fog and a fog level of one is maximum fog.</p>	<p>15 setFog 0.5</p>
<p>group setFormation formation Set group formation. Formation is one of:</p> <p>"COLUMN" "STAG COLUMN" "WEDGE"</p>	<p>groupOne setFormation "LINE"</p>

"ECH LEFT" "ECH RIGHT" "VEE" "LINE"	
object setFormationTask task Set the current task of the formation member.	
group setFormDir heading Set group formation heading. Accepted heading range is 0 to 360. Formation is facing this direction unless enemy is seen. When group is moving, this value is overridden by movement direction.	group1 setFormDir 180
side1 setFriend [side2, value] Sets how friendly side1 is with side2. For a value smaller than 0.6 it results in being enemy, otherwise it's friendly.	
vehicle setFuel amount Set fuel amount. Fuel 1 is full gas tank, 0 is empty.	jeepOne setFuel 0
vehicle setFuelCargo amount Set fuel amount in cargo space of refuel vehicle. Fuel 1 is full gas tank, 0 is empty.	refuelTruckOne setFuelCargo 0
group setGroupId [nameFormat, nameParam1, ...] Set group identity. Id format is [letter, color, picture] or [letter, color]. Letter is one of: "Alpha" "Bravo" "Charlie" "Delta" "Echo" "Foxtrot" "Golf" "Hotel" "Kilo" "Yankee" "Zulu" "Buffalo" "Convoy" "Guardian" "November" "Two" "Three" "Fox" Colour can be one of the following: "GroupColor0" - (Nothing) "GroupColor1" - Black "GroupColor2" - Red "GroupColor3" - Green	group1 setGroupId ["Delta", "GroupColor4"]

"GroupColor4" - Blue "GroupColor5" - Yellow "GroupColor6" - Orange "GroupColor7" - Pink "Six" - Six	
object setHideBehind [objectWhereHide, hidePosition] It sets the data for hiding. objectWhereHide can be taken using findCover. hidePosition can be taken using getHideFrom	
person setIdentity identity Set identity of person. Identities are defined in Description.ext of the mission or campaign.	soldier1 setIdentity "John_Doe"
light setLightAmbient [r, g, b] Set ambient color of light.	
light setLightBrightness brightness Set brightness of light.	
light setLightColor [r, g, b] Set diffuse color of light.	
name setMarkerBrush brush Selects the fill texture for the marker ("RECTANGLE" or "ELLIPSE"). Brush is the name of the subclass in CfgMarkerBrushes.	"Marker1" setMarkerBrush "DiagGrid"
name setMarkerBrushLocal brush Selects the fill texture for the marker ("RECTANGLE" or "ELLIPSE"). Brush is the name of the subclass in CfgMarkerBrushes.	"Marker1" setMarkerBrushLocal "DiagGrid"
marker setMarkerColor color Set marker color. Color is one of: "Default" "ColorBlack" "ColorRed" "ColorRedAlpha" "ColorGreen" "ColorGreenAlpha" "ColorBlue" "ColorYellow" "ColorWhite"	"MarkerOne" setMarkerColor "ColorBlack"
marker setMarkerColorLocal color Set marker color. Color is one of: "Default" "ColorBlack" "ColorRed" "ColorRedAlpha" "ColorGreen" "ColorGreenAlpha" "ColorBlue" "ColorYellow" "ColorWhite"	"MarkerOne" setMarkerColorLocal "ColorBlack"
name setMarkerDir angle Sets the orientation of the marker. Angle is in	"Marker1" setMarkerDir 90

degrees	
name setMarkerDirLocal angle Sets the orientation of the marker. Angle is in degrees.	"Marker1" setMarkerDirLocal 90
markerName setMarkerPos pos Moves the marker. Pos format is Position2D.	"MarkerOne" setMarkerPos getMarkerPos "MarkerTwo"
markerName setMarkerPosLocal pos Moves the marker. Pos format is Position2D.	"MarkerOne" setMarkerPos getMarkerPos "MarkerTwo"
handle setMarkerShape shape Selects the shape (type) of the marker. Shape can be "ICON", "RECTANGLE" or "ELLIPSE".	markerobj setMarkerShape "RECTANGLE"
handle setMarkerShapeLocal shape Selects the shape (type) of the marker. Shape can be "ICON", "RECTANGLE" or "ELLIPSE".	markerobj setMarkerShapeLocal "RECTANGLE"
marker setMarkerSize size Set marker size. Size is in format [a-axis, b-axis].	"MarkerOne" setMarkerSize [100, 200]
marker setMarkerSizeLocal size Set marker size. Size is in format [a-axis, b-axis].	"MarkerOne" setMarkerSizeLocal [100, 200]
name setMarkerText text Sets the text label of an existing marker.	"Marker1" setMarkerText "You are here."
name setMarkerTextLocal text Sets the text label of an existing marker.	"Marker1" setMarkerTextLocal "You are there."
markerName setMarkerType markertype Set marker type. Type may be any of: "Flag" "Flag1" "Dot" "Destroy" "Start" "End" "Warning" "Join" "Pickup" "Unknown" "Marker" "Arrow" "Empty"	"MarkerOne" setMarkerType "Warning"
markerName setMarkerTypeLocal markertype Set marker type. Type may be any of: "Flag" "Flag1" "Dot" "Destroy" "Start" "End" "Warning" "Join" "Pickup" "Unknown"	"MarkerOne" setMarkerTypeLocal "Warning"

"Marker" "Arrow" "Empty"	
person setMimic mimic Set person's facial expression. Following mimic values are recognized: "Default" "Normal" "Smile" "Hurt" "Ironic" "Sad" "Cynic" "Surprised" "Agresive" "Angry"	soldier1 setmimic "angry"
setMousePosition [x, y] Move (UI) mouse pointer to specified position of the screen. Values of x and y can be in the range from 0 to 1.	setMousePosition [0.5, 0.5]
trigger setMusicEffect track Defines the music track played on activation. Track is a subclass name of CfgMusic. "\$NONE\$" (no change) or "\$STOP\$" (stops the current music track). Alternativer Syntax: waypoint setMusicEffect track	trigger setMusicEffect "Track1"
object setObjectTexture texture Textures object selection with texture named in array. Array has the form [selection number, "Texture"]. The selection number is defined through the hiddenselection[]={ } array in the vehicle's config (starting with 0).	_objectname setobjecttexture [0, "\pboname\texture.paa"] or _objectname setobjecttexture [1, "\pboname\texture2.paa"]
time setOvercast overcast Set overcast to given value smoothly during given time (in seconds). Zero time means immediate change. Zero overcast means clear (sunny) weather, with overcast 1 storms and rain are very likely.	50 setOvercast 0.5
particleSource setParticleCircle [radius, velocity] Update particle source to create particles on circle with given radius. Velocity is transformed and added to total velocity.	
particleSource setParticleParams array Set parameters to particle source. Array is in format ParticleArray.	
particleSource setParticleRandom [lifeTime, position, moveVelocity, rotationVelocity, size, color, randomDirectionPeriod, randomDirectionIntensity]	

Set randomization of particle source parameters.	
setPlayable unit Create MP role for the unit. The roles created this way are used for Join In Progress and Team Switch.	setPlayable aP
obj setPos pos Set object position. Pos array format is Position.	player setPos [getPos player select 0, (getPos player select 1) + 10] or player setPos [getPos this select 0, getPos this select 1, (getPos this select 2) +10] or obj1 setpos [getPos obj1 select 0, getPos obj1 select 1, -5] Buries obj1 5 metres below ground level.
obj setPosASL pos Sets the object position. The pos array uses the PositionASL format.	player setPosASL [getPosASL player select 0, getPosASL player select 1 + 10, getPosASL select 2]
index setRadioMsg text Set radio message (0, 0, map radio) to given text. Use "NULL" to disable radio slot.	1 setRadioMsg "Alpha Radio"
time setRain rainDensity Set rain density smoothly during the given time (in seconds). A time of zero means an immediate change. A rain level of zero is no rain and a rain level of one is maximum rain Rain is not possible when overcast is smaller than 0.7.	60 setRain 1
unit setRank rank Sets rank of given unit. Possible values: PRIVATE, CORPORAL, SERGEANT, LIEUTENANT, CAPTAIN, MAJOR or COLONEL	player setRank "COLONEL"
vehicle setRepairCargo amount Set amount of repair resources in cargo space of repair vehicle. Amount 1 is full cargo.	repairTruck1 setRepairCargo 0
vehicle setSkill skill Sets ability level of person (commander unit). Value of skill may vary from 0 to 1.	hero setskill 1
vehicle setSkill skill Sets ability level of person (commander unit). Value of skill may vary from 0 to 1. Possible values are: aimingAccuracy aimingShake aimingSpeed endurance spotDistance spotTime courage reloadSpeed	hero setskill ["Endurance",0.7]

commanding general	
trigger setSoundEffect [sound, voice, soundEnv, soundDet] Defines the different sound effects. Sound / voice plays a 2D / 3D sound from CfgSounds. SoundEnv plays an enviromental sound from CfgEnvSounds. SoundDet (only for triggers) creates a dynamic sound object attached to a trigger defined in CfgSFX. Alternativer Syntax: waypoint setSoundEffect [sound, voice, soundEnv, soundDet]	trigger setSoundEffect ["Alarm", "", "", ""]
group setSpeedMode mode Set group speed mode. Mode may be one of: "LIMITED" (half speed) "NORMAL" (full speed, maintain formation) "FULL" (do not wait for any other units in formation)	groupOne setSpeedMode "LIMITED"
object setTargetAge age Sets how the target is known to the other centers. They behave like the target was seen age seconds ago. Possible age values are: "ACTUAL", "5 MIN", "10 MIN", "15 MIN", "30 MIN", "60 MIN", "120 MIN" or "UNKNOWN".	player setTargetAge "10 MIN"
setTerrainGrid grid Operation Flashpoint, VBS1: Set desired terrain resolution (in meters). For default landscapes, supported resolutions are: 50 - smoothest, less lag 25 - default in multiplayer 12.5 - default in singleplayer 6.25 3.125 - bumpiest, higher lag If you select unsupported resolutions, nearest supported value is used instead. Armed Assault, VBS2: Terrain resolution is fixed, determined by the world created. This function controls terrain LOD instead (the distance in which the terrain mesh resolution starts to degrade). Higher number means less vertices are used for terrain rendering, making distant hills less smooth. Value 12.5 corresponds to selecting Terrain Detail Normal in Video options, 50 to Very Low, 3.125 to Very High.	setTerrainGrid 12.5
trigger or waypoint setTitleEffect [type, effect, text] Defines the title effect.	trigger setTitleEffect ["TEXT", "PLAIN DOWN", "Hello world."]

<p>Type can be "NONE", "OBJECT", "RES" or "TEXT".</p> <p>For "TEXT", the effect defines a subtype: "PLAIN", "PLAIN DOWN", "BLACK", "BLACK FADED", "BLACK OUT", "BLACK IN", "WHITE OUT" or "WHITE IN".</p> <p>Text is shown as text itself.</p> <p>For "OBJECT", text defines the shown object , a subclass of CfgTitles.</p> <p>For "RES", text defines a resource class, a subclass of RscTitles.</p>	
<p>trigger setTriggerActivation [by, type, repeating]</p> <p>Defines the trigger activation type.</p> <p>First argument - who activates trigger (side, radio, vehicle or group member): "NONE", "EAST", "WEST", "GUER", "CIV", "LOGIC", "ANY", "ALPHA", "BRAVO", "CHARLIE", "DELTA", "ECHO", "FOXTROT", "GOLF", "HOTEL", "INDIA", "JULIET", "STATIC", "VEHICLE", "GROUP", "LEADER" or "MEMBER".</p> <p>Second argument - when is it activated (presentation or detection by the specified side): "PRESENT", "NOT PRESENT", "WEST D", "EAST D", "GUER D" or "CIV D".</p> <p>Third argument - whether the activation is repeating.</p>	<p>trigger setTriggerActivation ["WEST", "EAST D", true]</p>
<p>trigger setTriggerArea [a, b, angle, rectangle]</p> <p>Defines the area controlled by the trigger. The area is rectangular or elliptic, the width is 2 * a, the height is 2 * b. It is rotated angle degrees.</p>	<p>trigger setTriggerArea [100, 50, 45, false]</p>
<p>trigger setTriggerStatements [cond, activ, desactiv]</p> <p>The first argument can modify the condition of when the trigger is activated. The result of the activation defined by trigger activation is in variable this. Variable thisList contains all vehicles which caused the activation. Activ and desactiv expressions are launched upon trigger activation / deactivation.</p>	<p>trigger setTriggerStatements ["this", "ok = true", "ok = false"]</p>
<p>trigger setTriggerText text</p> <p>Sets the text label attached to the trigger object. This is used for example as a radio slot label for radio activated triggers.</p>	<p>trigger setTriggerText "Call for support"</p>
<p>trigger setTriggerTimeout [min, mid, max, interruptable]</p> <p>Defines the time between condition satisfaction and trigger activation (randomly from min to max, with an average value mid). If the last argument is true, the condition must be fulfilled all the time.</p>	<p>trigger setTriggerTimeout [5, 10, 7, false]</p>

<p>trigger setTriggerType action Sets the type of action processed by the trigger after activation (no action, a waypoints switch or an end of mission): "NONE", "SWITCH", "END1", "END2", "END3", "END4", "END5", "END6", "LOOSE" or "WIN".</p>	trigger setTriggerType "END1"
<p>unit setUnitAbility skill Sets skill of given unit. This command will probably differ in some future products, but currently it does the same as setSkill. Skill may vary from 0.2 to 1.0.</p>	player setUnitAbility 1.0
<p>unit setUnitPos mode Set unit position rules. Mode may be one of: "DOWN" - person goes prone and stays prone. "UP" - person stands and stays standing. "Middle" - Kneel Position. ArmA version 1.04 "AUTO" - person chooses mode according to circumstances</p>	loon1 setUnitPos "UP"
<p>unit setUnitPosWeak mode Set unit position rules. Mode may be one of: "DOWN" - person goes prone and stays prone. "UP" - person stands and stays standing. "Middle" - Kneel Position. ArmA version 1.04 "AUTO" - person chooses mode according to circumstances</p>	
<p>unit setUnitRank rank Sets rank of given unit. Possible values: PRIVATE, CORPORAL, SERGEANT, LIEUTENANT, CAPTAIN, MAJOR or COLONEL.</p>	player setUnitRank "COLONEL"
<p>object setVariable [name, value] Set variable to given value in the variable space of given object. The object must be a vehicle, otherwise the function does nothing.</p>	myTruck setVariable ["myVariable",123]
<p>object setVectorDir [x, z, y] Set object's direction vector. Up vector will remain unchanged.</p>	
<p>object setVectorUp [x, z, y] Set object's up vector. Direction vector will remain unchanged.</p>	
<p>object setVehicleAmmo value Sets how much ammunition (compared to a full state defined by the vehicle type) the vehicle has. The value ranges from 0 to 1.</p>	player setVehicleAmmo 0
<p>object setVehicleArmor value Sets the armor (or health for men) state of the vehicle (a value from 0 to 1).</p>	player setVehicleArmor 0.5
<p>object setVehicleId id Sets id (integer value) to vehicle. By this id</p>	player setVehicleId 1

vehicle is referenced by triggers and waypoints.	
vehicle setVehicleInit statement Attach a statement to a vehicle. The statement is propagated over the network in MP games, it can be executed by invoking processInitCommands.	soldier3 setVehicleInit "this allowfleeing 0"
vehicle setVehicleLock state Set how vehicle is locked for player. Possible values: "UNLOCKED", "DEFAULT" or "LOCKED".	veh1 setVehicleLock "LOCKED"
object setVehiclePosition [position, markers, placement] Changes the object position. If the markers array contains more than one marker name, the position of a random one is used. Otherwise, the given position is used. The object is placed inside a circle with this position as its center and placement as its radius.	player setVehiclePosition [[0, 0, 0], ["Marker1"], 0]
object setVehicleVarName name Sets the name of the variable which contains a reference to this object. It is necessary in MP to change the variable content after a respawn.	player setVehicleVarName "aP"
vehicle setVelocity [x, y, z] Set velocity (speed vector) of vehicle.	truck1 setVelocity [20, 0, 0] or Advanced method used for relative acceleration: _vel = velocity _vehicle; _dir = direction _vehicle; _speed = 10; comment "Added speed"; _vehicle setVelocity [(_vel select 0)+(sin _dir*_speed),(_vel select 1)+(cos _dir*_speed),(_vel select 2)];
setViewDistance distance Set rendering distance, in metres. Default is 900m, accepted range is 500m to 5000m.	setViewDistance 2250
waypoint setWaypointBehaviour mode Switches the unit behaviour when the waypoint becomes active. Possible values are: "UNCHANGED", "CARELESS", "SAFE", "AWARE", "COMBAT" and "STEALTH"	[grp, 2] setWaypointBehaviour "AWARE"
waypoint setWaypointCombatMode mode The group combat mode is switched when the waypoint becomes active. Possible values are: "NO CHANGE", "BLUE", "GREEN", "WHITE", "YELLOW" and "RED".	[grp, 2] setWaypointCombatMode "RED"
waypoint setWaypointDescription text Sets the description shown in the HUD while the waypoint is active.	[grp, 2] setWaypointDescription "Move here."
waypoint setWaypointFormation formation Switches the group formation when the waypoint becomes active.	[grp, 2] setWaypointFormation "LINE"

Possible values are: "NO CHANGE", "COLUMN", "STAG COLUMN", "WEDGE", "ECH LEFT", "ECH RIGHT", "VEE", "LINE", "FILE" and "DIAMOND".	
waypoint setWaypointHousePosition pos For waypoints attached to a house, this defines the target house position.	[grp, 2] setWaypointHousePosition 1
waypoint setWaypointPosition [center, radius] Moves the waypoint to a random position in a circle with the given center and radius.	[grp, 2] setWaypointPosition [position player, 0]
waypoint setWaypointScript command Attaches a script to a scripted waypoint. Command consist of a script name and additional script arguments.	[grp, 2] setWaypointScript "find.sqs player"
waypoint setWaypointSpeed mode Switches the group speed mode when the waypoint becomes active. Possible values are: "UNCHANGED", "LIMITED", "NORMAL" and "FULL".	[grp, 2] setWaypointSpeed "FULL"
waypoint setWaypointStatements [condition, statement] The waypoint is done only when the condition is fulfilled. When the waypoint is done, the statement expression is executed.	[grp, 2] setWaypointStatements ["true", ""]
waypoint setWaypointTimeout [min, mid, max] Defines the time between condition satisfaction and waypoint finish (randomly from min to max, with an average value mid).	[grp, 2] setWaypointTimeout [5, 10, 6]
waypoint setWaypointType type Changes the waypoint type. Type can be: "MOVE", "DESTROY", "GETIN", "SAD", "JOIN", "LEADER", "GETOUT", "CYCLE", "LOAD", "UNLOAD", "TR UNLOAD", "HOLD", "SENTRY", "GUARD", "TALK", "SCRIPTED", "SUPPORT", "GETIN NEAREST", "AND" or "OR".	[grp, 2] setWaypointType "HOLD"
waypoint setWPPos position Set waypoint position. Waypoint is in format Waypoint. Position is in format Position2D.	[groupOne, 1] setWPPos markerPos "MarkerOne"
showCinemaBorder show Force drawing of cinema borders. This is normally used in cutscenes to indicate player has no control.	showCinemaBorder true
showCompass show Enable compass (default true)	showCompass false
showGPS show Enable GPS receiver (default false)	showGPS true
showMap show Enable Map (default true)	showMap false
shownCompass	? showncompass : hint "You have a

Check if player has compass enabled.	compass."
shownGPS Check if player has GPS receiver enabled.	? shownGPS : hint "You have a GPS receiver."
shownMap Check if player has Map enabled.	? shownMap : hint "You have a Map."
shownPad Check if player has Notebook enabled.	? shownPad : hint "You have a Notebook."
shownRadio Check if player has Radio enabled.	? shownRadio : hint "You have a Radio."
shownWarrant Check if player has ID card enabled. Obsolete command.	
shownWatch Check if player has Watch enabled.	? shownWatch : hint "You have a Watch."
showPad show Enable Notebook (default true)	showPad false
showRadio show Enable Radio (default true)	showRadio false
showWarrant show Enable ID card (default false). Obsolete command.	
showWatch show Enable Watch (default true)	showWatch false
waypoint show Waypoint show Sets the condition determining when the waypoint is shown. Possible values are: "NEVER", "EASY" and "ALWAYS".	[grp, 2] showWaypoint "ALWAYS"
side unit Returns the side of unit. Once dead, a unit will be on the civilian side. When used in conjunction with a format statement (hint format["%1",side player]), the returned strings are: "WEST", "EAST", "GUER" or "CIV".	? (side player == west) : hint "You are on the West side."
unit sideChat chatText Types text to the side radio channel. Note: This function only types text to the list, it does not broadcast the message. If you want the message to show on all computers, you have to execute it on all of them.	soldierOne sideChat "Show this text"
sideEnemy The Enemy side (used for renegades).	
sideFriendly The Friendly side (used for captives).	
sideLogic The Logic side.	
unit sideRadio radioName Send the message to the side radio channel. Message is defined in Description.ext file.	soldierOne sideRadio "messageOne"
sin x	_sine = sin 30

Sine of x, argument in degrees.	Result is 0.5
skill person Returns current level of ability of person, in range between 0 and 1. Skill 1 is highest skill.	_sk = skill loon1
vehicle skill type Returns current level of ability of person, in range between 0 and 1. Skill 1 is highest skill. Type can be one of: aimingAccuracy aimingShake aimingSpeed endurance spotDistance spotTime courage reloadSpeed commanding general	_sk = skill loon1
skipTime duration Skip time in duration of hours. Daytime is adjusted, weather change is estimated, no changes in any units are made. The tide is also adjusted.	skipTime 5
sleep delay Suspend execution of Function or SQF Script for given time.	sleep 0.5
sliderPosition idc Return current thumb position of slider idc of topmost user dialog.	_slidepos1 = sliderPosition 105
sliderPosition control Returns the current thumb position of the given slider.	_slidepos1 = sliderPosition _control
sliderRange idc Return limits, as an Array [min, max] of slider idc of topmost user dialog.	_slidelimits1 = sliderRange 105
sliderRange control Returns the limits (as an array [min, max]) of the given slider.	_slidelimits1 = sliderRange _control
sliderSetPosition [idc,pos] Set current thumb position of slider idc of topmost user dialog.	sliderSetPosition [101, 50]
control sliderSetPosition pos Sets the current thumb position of the given slider.	sliderSetPosition [101, 50]
sliderSetRange [idc,min,max] Set limits of slider idc of topmost user dialog.	sliderSetRange [101, 0, 100]
control sliderSetRange [min,max] Sets the limits of the slider with id idc of the given slider.	sliderSetRange [101, 0, 100]
sliderSetSpeed [idc,line,page] Set speed of slider with id idc of topmost user	sliderSetspeed [101, 0.5, 2]

dialog. Click to arrow = move by line Click to scale outside thumb = move by page.	
control sliderSetSpeed [line,page] Sets the speed of the given slider. Click to arrow - move by line Click to scale outside thumb - move by page.	sliderSetspeed [101, 0.5, 2]
sliderSpeed idc Return speed, as an Array [min, max] of slider idc of topmost user dialog.	_slidespeed1 = sliderspeed 105
sliderSpeed control Returns the speed (as an Array [line, page]) of the given slider.	_slidespeed1 = sliderspeed 105
someAmmo unit Check if unit has some ammo.	? not (someAmmo loon1) : hint "Loon1 is out of ammo!"
soundVolume Check current sound volume (set by fadeSound).	_vol = soundVolume
spawn code Starts running a new script. The new script is running in parallel, spawn does not wait for it to be done. To check if it finished use scriptDone	
arguments spawn code Starts running a new script (Function). Additional arguments are passed in local _this variable. The new script (Function) is running in parallel, spawn does not wait for it to be done. To check if it finished use scriptDone	
speed obj Object speed (in km/h).	? (speed truck1) >= 100 : hint "You're going too fast!"
speedMode grp Returns speed mode of the group, which can be any of the following: "LIMITED" "NORMAL" "FULL"	_grpspeed1 = speedMode grp1
sqrt x Returns square root of x.	_sq = sqrt 9 Result is 3
for /.../ step step Optionally can set step. If you want to count down, step must be specified, and set negative. Default value is 1.	for "_x" from 20 to 10 step -2 do {..code..}
unit stop stop Stop AI unit. Stopped unit will not be able to move and fire. Use disableAI to choose which part of AI you want to stop	loon1 stop true
stopped unit Check if unit is stopped by stop command.	? (stopped loon1) : hint "Loon1 is stopped"
str any value Converts any variable to a string.	str(2+3) , result is "5"
supportInfo mask Creates a list of supported operators and type. Each field of array has the format: "x:name"	supportInfo "b:select*" result is ["b:ARRAY select SCALAR","b:ARRAY select

<p>Where x can be: 't' - type 'n' - null operator 'u' - unary operator 'b' - binary operator. 'name' is the operator or type name (in case operator, type of input operands is included). `mask` parameter can be an empty string, or one of field. In this case, function returns empty array, if operator is not included in the list. `mask` can contain wildcards, for example: *:name, t:*, t:name* or *:*. </p>	<p>BOOL"]</p>
<p>surfaceIsWater [x, y] Returns whether water is at given position.</p>	
<p>surfaceType [x, y] Returns what surface is at the given position.</p>	
<p>switch exp See switch do</p>	<pre>switch (_a) do { case 1: {block}; case 2 : {block}; default {block}; } or _color=switch (side player) do { case west: {"ColorGreen"}; case east: {"ColorRed"}; };</pre>
<p>switchableUnits Return a list of units accessible through Team Switch.</p>	
<p>unit switchCamera mode Switch camera to given vehicle / camera. Mode is one of: "INTERNAL" (1st person) "GUNNER" (optics / sights) "EXTERNAL" (3rd person) "GROUP" (group)</p>	<pre>loon1 switchCamera "External"</pre>
<p>lamppost switchLight mode Controls lamppost mode. Mode may be: "ON" "OFF" "AUTO" (Lampost is on only during nighttime) "AUTO" is default.</p>	<pre>(object 12345) switchLight "off" nearestObject [player, "Streetlamp"] switchLight "OFF"</pre>
<p>soldier switchmove movename When used on a person, the given move is started immediately (there is no transition). Use switchmove "" to switch back to the default movement if there is no transition back, otherwise the person may be stuck.</p>	<pre>loon1 switchMove "FXStandDip"</pre>
<p>waypoint synchronizeWaypoint [waypoint1, waypoint2, ...] Synchronizes the waypoint with other waypoints. Each waypoint is given as an array [group, index].</p>	<pre>[group1, 2] synchronizeWaypoint [[group2, 3]]</pre>
<p>waypoint synchronizeWaypoint [waypoint1, waypoint2, ...]</p>	<pre>[group1, 2] synchronizeWaypoint [[group2, 3]]</pre>

Synchronizes the waypoint with other waypoints. Each waypoint is given as an array [group, index].	
tan x Tangent of x, argument in degrees	_tangent = tan 45 Ergebnis: 1
team switch Invoke the team switch dialog (force it even when conditions are not met).	
teamSwitchEnabled Check if team switch is currently enabled.	
terminate Terminate (abort) the script	_Script=[] ExecVM "Script.sqs"; Sleep 5; Terminate _Script; Hint "Script.sqs has been terminated after 5 seconds";
text text Creates a structured text containing the given plain text.	txt2 = text "Hello world."
textLog anything Dump argument value to debugging output. Note: This command has no real use in the retail version, and does nothing.	textLog player
tg x Tangent of x, argument in degrees.	_tangent = tg 45Result is 1
if then else First or second element of array is executed depending on result of <u>if</u> condition. Result of the expression executed is returned as a result (result may be nothing).	if (a>b) then {c=1} else {c=2} or if (a>b) then {c=1;c=2}
throw expression Throws an exception. The exception is processed by first catch block.	throw "invalid argument"
time Time elapsed since mission started (in seconds).	
titleCut effect Obsolete command.	
titleObj effect Object title - argument in format ["text", "type", speed] or ["name", "type"] If speed is not given, it is assumed 1. Object can be defined in Description.ext	titleObj ["BISLogo", "PLAIN"]
titleRsc effect Resource title - argument in format ["name", "type", speed] or ["name", "type"] If speed is not given, it is assumed 1. Resource can be defined in Description.ext Also see cutRsc, with these two commands you can show two different resources at once.	titleRsc ["BIS", "PLAIN"]
titleText effect Text title - argument in format ["text", "type", speed] or ["text", "type"] If speed is not given, it	titleText ["Show this text", "PLAIN"]

is assumed 1. Also see cutText, with these two commands you can show two different texts at once (optimally with two different types).	
for "_var" from a to b Continue sequence of 'for' command.	for "_x" from 10 to 20 do {..code..}
trigger triggerAttachObject objectId Assigns a static object to the trigger. The activation source is changed to "STATIC".	trigger triggerAttachObject 1234
trigger triggerAttachVehicle [] or [vehicle] Attach a trigger to a vehicle. If [] is given, the trigger is detached from the assigned vehicle. If the activation source is "VEHICLE", "GROUP", "LEADER" or "MEMBER", it's changed to "NONE". If [vehicle] is given, the trigger is attached to the vehicle or its group. When the source is "GROUP", "LEADER" or "MEMBER", it's attached to the group, otherwise it's attached to the vehicle and the source is changed to "VEHICLE".	trigger triggerAttachVehicle [player]
true Always true	
try code Defines a try-catch structure. This sets up an exception handling block. Any thrown exception in a try block is caught in a catch block. The structured exception block has following form try //begin of try-catch block { //block, that can throw exception } catch{ //block, that process an exception. Exception is described in _exception variable };	
typeName any Returns the data type of expression. Type is returned as string	_x="hello"; typeName _x, result is "string" or _x=player; typeName _x, result is "object"
typeOf vehicle Returns the class type of a given object or vehicle.	_class = typeOf _mi24
unassignTeam vehicle Unassigns the vehicle (its commander unit) from his team. This is equal to vehicle assignTeam "MAIN".	unassignTeam soldier2
unassignVehicle unit Unit is unassigned from the vehicle. If he is currently in, group leader will issue order to disembark. Examples: unassignVehicle player will make the player disembark {unassignVehicle	unassignVehicle player

<code>_x }</code> forEach crew vehiclename will make all the occupants of a vehicle disembark	
unitPos person Return the unit position rules.	
unitReady unit Check if the unit is ready. Unit is busy when it is given some command like move, until the command is finished.	<code>_it = unitReady soldierOne</code>
units grp Returns an array with all the units in the group.	<code>player in units group player</code>
units unit Returns an array with all the units in the group of the given object. For a destroyed object an empty array is returned.	<code>player in units player</code>
vectorDir obj Return object's direction vector in world coordinates as [x, z, y]. A unit facing North would return [0,1,0] A unit facing East would return [1,0,0] A unit facing South would return [0,-1,0] A unit facing West would return [-1,0,0]	
vectorUp obj Return object's up vector in world coordinates as [x, y, z].	
vehicle unit Vehicle in which given unit is mounted. If none, unit is returned.	<code>? vehicle player != player : hint "Player is in a vehicle"</code>
vehicleChat Type text to vehicle radio channel. Note: This function only types text to the list, it does not broadcast the message. If you want the message to show on all computers, you have to execute it on them.	<code>soldierOne vehicleChat "Show this text"</code>
vehicleRadio Send message to vehicle radio channel. Message is defined in description.ext.	<code>soldierOne vehicleRadio "messageOne"</code>
vehicles Return a list of vehicles in the current mission, including soldiers. It does not list soldiers which are boarded in a vehicle.	<code>_vehicles = vehicles</code>
vehicleVarName object Returns the name of the variable which contains a primary editor reference to this object. This is the variable given in the Insert Unit dialog / name field, in the editor. It can be changed using <code>setVehicleVarName</code> .	
velocity vehicle Return velocity (speed vector) of vehicle as array [x, z, y].	<code>vector = velocity jeep</code> or <code>? velocity carOne > 50 : hint "Slow down, you are exceeding the speed limit."</code>

verifySignature filename Check if file is signed by an accepted key.	
waitUntil condition Suspend execution of function or SQF based script until condition is satisfied.	waitUntil {not alive player} or _i = 0; waitUntil {_i = _i + 1; _i >= 100}
waypoint waypointAttachObject idStatic Attaches a static object to the given waypoint.	[grp, 2] waypointAttachObject 1234
waypoint waypointAttachVehicle vehicle Attaches a vehicle to the given waypoint.	[grp, 2] waypointAttachVehicle vehicle player
waypointPosition waypoint Get waypoint position. Waypoint format is Waypoint. Note: This function is identical to getWPPos.	wPos = waypointPosition [groupOne, 1]
vehicle weaponDirection weaponName Returns the direction that the given weapon is pointing at.	_dir = _vehicle weaponDirection "M16"
weapons vehicle Returns array of names of all vehicle's weapons.	wArray = weapons player
west West side	
while condition First part of while construct	while {x<10} do {x=x+1} or A practical example: Repair all members of a group to such a level that they are able to stand up: { while {(not canStand _x) and (alive _x)} do { _x setDammage (getDammage _x) - 0.01; };} forEach units group unitname;
Wind Returns the current wind vector as array [x, z, y].	
worldName Return the name of the currently loaded world.	_name = worldName
object worldToModel worldPos Converts position from world space to object model space.	

Animationsliste

Hier sind alle Animationsbefehle aufgezigt. Es handelt sich hierbei um eine Liste der BIWiki. Animationen ab zu stimmen ist eine echte Kunst.

unit [switchMove](#) move

(Table code: "Usage:s")

unit [playMove](#) move

(Table code: "Usage:p")

Acrg...

Name	Usage	Description
AcrgPknLMstpSnonWnonDnon_AmovPercMstpSnonWnonDnon_getOutHigh	-	-
AcrgPknLMstpSnonWnonDnon_AmovPercMstpSnonWnonDnon_getOutLow	-	-
AcrgPknLMstpSnonWnonDnon_AmovPercMstpSnonWnonDnon_getOutMedium	-	-
AcrgPknLMstpSnonWnonDnon_AmovPercMstpSrasWpstDnon_getOutHigh	-	-
AcrgPknLMstpSnonWnonDnon_AmovPercMstpSrasWpstDnon_getOutLow	-	-
AcrgPknLMstpSnonWnonDnon_AmovPercMstpSrasWpstDnon_getOutMedium	-	-
AcrgPknLMstpSnonWnonDnon_AmovPercMstpSrasWrflDnon_getOutHigh	-	-
AcrgPknLMstpSnonWnonDnon_AmovPercMstpSrasWrflDnon_getOutLow	-	-
AcrgPknLMstpSnonWnonDnon_AmovPercMstpSrasWrflDnon_getOutMedium	-	-

Acts...

Name	Usage	Description
ActsPercMstpSnonWnonDnon_MarianQ_TVstudioMan	p	TV Studio Man animation
ActsPercMstpSnonWnonDnon_MarianQ_TVstudioMan_Loop1	p	TV Studio Man animation
ActsPercMstpSnonWnonDnon_MarianQ_TVstudioMan_Loop2	p	TV Studio Man animation
ActsPercMstpSnonWnonDnon_MarianQ_TVstudioMan_Loop3	p	TV Studio Man animation
ActsPercMstpSnonWnonDnon_MarianQ_TVstudioMan_LoopLong	p	TV Studio Man animation
ActsPercMstpSnonWnonDnon_MarianQ_WarReporter	p	Interview scene (strange gfx)
ActsPercMstpSnonWnonDnon_MarianQ_shot1	p	Interview scene (strange gfx)
ActsPercMstpSnonWnonDnon_MarianQ_shot1Man	p	Interview partner holding rifle
ActsPercMstpSnonWnonDnon_MarianQ_shot2	p	Interview scene (strange gfx)
ActsPercMstpSnonWnonDnon_MarianQ_shot3	p	Interview scene (strange gfx)
ActsPercMstpSnonWnonDnon_MarianQ_shot3Man	p	Interview partner listening, nothing in hand
ActsPercMstpSnonWnonDnon_MarianQ_shot4	p	Interview scene (strange gfx)
ActsPercMstpSnonWnonDnon_MarianQ_shot4Man	p	Interview partner holding rifle, give orders, moves some steps
ActsPercMstpSnonWnonDnon_MarianQ_shot5	p	Interview scene (strange gfx)
ActsPercMstpSnonWnonDnon_MarianQ_shot5Man	p	Interview partner holding rifle, give quick orders, moves and prepares to defend

Adth...

Name	Usage	Description
AdthPercMrunSlowWlnrDb_1	p	knees, takes launcher from back, dies
AdthPercMrunSlowWlnrDf_1	p	knees, takes launcher from back, stands up, dies
AdthPercMrunSlowWlnrDf_2	p	knees, takes launcher from back, stands up, dies
AdthPercMrunSlowWrflDf_6	p	knees, takes rifle from back, dies
AdthPercMstpSlowWlnrDnon_1	p	knees, takes launcher form back, stands up, few steps forward, dies falling left
AdthPercMstpSlowWlnrDnon_2	p	knees, takes launcher from back, stands up, few steps forward, dies falling right
AdthPercMstpSlowWrflDf_1	p	takes rifle from back, dies falling left
AdthPercMstpSlowWrflDf_2	p	takes rifle from back, dies falling right

AdthPercMstpSlowWrflDf_4	p	takes rifle from back, dies falling forward
AdthPercMstpSlowWrflDnon_1	p	takes rifle from back, dies falling forward
AdthPercMstpSlowWrflDnon_2	p	takes rifle from back, dies falling backward
AdthPercMstpSlowWrflDnon_4	p	takes rifle from back, dies falling backward right
AdthPercMstpSlowWrflDnon_8	p	takes rifle from back, dies falling forward left
AdthPercMstpSlowWrflDnon_binocular	p	stands, switches to binocular, then dies
AdthPercMstpSnonWnonDnon_1	p	dies, falling forward and turns on back
AdthPercMstpSnonWnonDnon_2	p	dies, falling backward and turns on back
AdthPercMstpSnonWnonDnon_3	p	dies, falling forward
AdthPercMstpSnonWnonDnon_binocular	p	dies, when taking binocular
AdthPercMstpSrasWpstDb_8	p	takes pistol, dies falling forward
AdthPercMstpSrasWpstDb_9	p	takes pistol, dies falling backward
AdthPercMstpSrasWpstDf_2	p	takes pistol, dies falling forward
AdthPercMstpSrasWpstDf_6	p	takes pistol, dies falling forward
AdthPercMstpSrasWpstDf_7	p	takes pistol, dies falling forward
AdthPercMstpSrasWpstDnon_1	p	takes pistol, dies falling forward
AdthPercMstpSrasWpstDnon_3	p	takes pistol, dies falling forward left
AdthPercMstpSrasWpstDnon_5	p	takes pistol, dies falling forward
AdthPercMstpSrasWrflDb_8	p	takes rifle, dies falling forward
AdthPercMstpSrasWrflDf_1	p	takes rifle, dies falling forward
AdthPercMstpSrasWrflDf_2	p	takes rifle, dies falling forward
AdthPercMstpSrasWrflDf_4	p	takes rifle, dies falling forward
AdthPercMstpSrasWrflDnon_1	p	holds rifle, dies falling forward
AdthPercMstpSrasWrflDnon_2	p	holds rifle, dies falling forward left
AdthPercMstpSrasWrflDnon_4	p	holds rifle, dies falling forward
AdthPercMstpSrasWrflDnon_8	p	holds rifle, dies falling forward on back
AdthPknIMstpSlayWrflDnon_inventory	p	unholsters pistol, kneels down (inventory check), dies
AdthPknIMstpSlowWlnrDnon_binocular	-	-
AdthPknIMstpSlowWrflDnon_binocular	p	Kneels; switches to binocular, then dies
AdthPknIMstpSnonWnonDnon_1	-	-
AdthPknIMstpSnonWnonDnon_2	-	-
AdthPknIMstpSnonWnonDnon_binocular	-	-
AdthPknIMstpSnonWnonDnon_inventory	p	unholsters pistol, kneels down (inventory check), dies
AdthPknIMstpSrasWlnrDnon_1	-	-
AdthPknIMstpSrasWlnrDnon_2	-	-
AdthPknIMstpSrasWpstDb_5	-	-
AdthPknIMstpSrasWpstDnon_1	-	-
AdthPknIMstpSrasWpstDnon_2	-	-
AdthPknIMstpSrasWpstDnon_4	-	-
AdthPknIMstpSrasWpstDnon_6	-	-
AdthPknIMstpSrasWrflDnon_1	-	-
AdthPknIMstpSrasWrflDnon_2	-	-
AdthPknIMwlkSrasWlnrDf_1	-	-
AdthPknIMwlkSrasWlnrDf_2	-	-
AdthPknIMwlkSrasWrflDf_1	-	-
AdthPknIMwlkSrasWrflDf_2	-	-
AdthPpneMstpSlowWrflDf_1	-	-
AdthPpneMstpSlowWrflDf_2	-	-
AdthPpneMstpSlowWrflDnon_binocular	-	-
AdthPpneMstpSnonWnonDnon	-	-
AdthPpneMstpSnonWnonDnon_binocular	-	-
AdthPpneMstpSrasWpstDnon_1	-	-
AdthPpneMstpSrasWpstDnon_2	-	-
AdthPpneMstpSrasWrflDnon_1	-	-

AdthPpneMstpSrasWrflDnon_2	-	-
AdthPsitMstpSlowWrflDnon	p	Sitting; dies
AdthPswmMrunSnonWnonDf	-	-
AdthPswmMstpSnonWnonDnon	-	-

Aidl..

Name	Usage	Description
AidlPercMstpSlowWrflDnon01	-	-
AidlPercMstpSlowWrflDnon02	-	-
AidlPercMstpSlowWrflDnon03	-	-
AidlPercMstpSlowWrflDnon04	-	-
AidlPercMstpSlowWrflDnon05	-	-
AidlPercMstpSlowWrflDnon06	-	-
AidlPercMstpSlowWrflDnon0S	-	-
AidlPercMstpSnonWnonDnon01	-	-
AidlPercMstpSnonWnonDnon02	-	-
AidlPercMstpSnonWnonDnon03	-	-
AidlPercMstpSnonWnonDnon04	p	Puts weapon on back
AidlPercMstpSnonWnonDnon05	-	-
AidlPercMstpSnonWnonDnon06	-	-
AidlPercMstpSnonWnonDnon07	-	-
AidlPercMstpSnonWnonDnon08	-	-
AidlPercMstpSnonWnonDnon0S	-	-
AidlPercMstpSrasWpstDnon01	-	-
AidlPercMstpSrasWpstDnon02	-	-
AidlPercMstpSrasWpstDnon03	-	-
AidlPercMstpSrasWpstDnon0S	-	-
AidlPercMstpSrasWrflDnon01	-	-
AidlPercMstpSrasWrflDnon02	-	-
AidlPercMstpSrasWrflDnon03	-	-
AidlPercMstpSrasWrflDnon04	-	-
AidlPercMstpSrasWrflDnon05	-	-
AidlPercMstpSrasWrflDnon06	-	-
AidlPercMstpSrasWrflDnon0S	-	-
AidlPknlMstpSnonWnonDnon01	p	Combat animation
AidlPknlMstpSnonWnonDnon02	-	-
AidlPknlMstpSnonWnonDnon03	-	-
AidlPknlMstpSnonWnonDnon0S	-	-
AidlPknlMstpSrasWlnrDnon01	-	-
AidlPknlMstpSrasWlnrDnon02	-	-
AidlPknlMstpSrasWlnrDnon03	-	-
AidlPknlMstpSrasWlnrDnon0S	-	-
AidlPknlMstpSrasWpstDnon01	-	-
AidlPknlMstpSrasWpstDnon02	-	-
AidlPknlMstpSrasWpstDnon03	-	-
AidlPknlMstpSrasWpstDnon0S	-	-
AidlPknlMstpSrasWrflDnon01	-	-
AidlPknlMstpSrasWrflDnon02	-	-
AidlPknlMstpSrasWrflDnon03	-	-
AidlPknlMstpSrasWrflDnon0S	-	-
AidlPpneMstpSnonWnonDnon01	-	-
AidlPpneMstpSnonWnonDnon02	-	-
AidlPpneMstpSnonWnonDnon03	-	-
AidlPpneMstpSnonWnonDnon0S	-	-

AidlPpneMstpSrasWpstDnon01	-	-
AidlPpneMstpSrasWpstDnon02	-	-
AidlPpneMstpSrasWpstDnon03	-	-
AidlPpneMstpSrasWpstDnon0S	-	-
AidlPpneMstpSrasWrflDnon01	-	-
AidlPpneMstpSrasWrflDnon02	-	-
AidlPpneMstpSrasWrflDnon03	-	-
AidlPpneMstpSrasWrflDnon0S	-	-
AidlRflflowToRasTrans	-	-
AidlRflrasToLowTrans	-	-

Ainv...

Name	Usage	Description
AinvPknIMstpSlayWrflDnon	-	-
AinvPknIMstpSlayWrflDnon_1	-	-
AinvPknIMstpSlayWrflDnon_AmovPercMstpSnonWnonDnon	-	-
AinvPknIMstpSlayWrflDnon_AmovPercMstpSrasWrflDnon	p	Kneels in front of ammo box, then gets up again.
AinvPknIMstpSlayWrflDnon_AmovPknIMstpSnonWnonDnon	-	-
AinvPknIMstpSlayWrflDnon_AmovPknIMstpSrasWrflDnon	p	Head injury/ache
AinvPknIMstpSlayWrflDnon_healed	p	Unit being healed
AinvPknIMstpSlayWrflDnon_healed2	-	-
AinvPknIMstpSlayWrflDnon_medic	p	Medic healing
AinvPknIMstpSnonWnonDnon_1	p	Kneels in front of ammo box. Stays that way
AinvPknIMstpSnonWnonDnon_2	-	-
AinvPknIMstpSnonWnonDnon_3	-	-
AinvPknIMstpSnonWnonDnon_4	-	-
AinvPknIMstpSnonWnonDnon_AmovPercMstpSrasWpstDnon	-	-
AinvPknIMstpSnonWnonDnon_AmovPercMstpSrasWpstDnon_end	-	-
AinvPknIMstpSnonWnonDnon_AmovPknIMstpSrasWpstDnon	p	Head injury/ache
AinvPknIMstpSnonWnonDnon_AmovPknIMstpSrasWpstDnon_end	-	-
AinvPknIMstpSnonWnonDnon_healed_1	-	-
AinvPknIMstpSnonWnonDnon_healed_2	-	-
AinvPknIMstpSnonWnonDnon_medic_1	-	-
AinvPknIMstpSnonWnonDnon_medic_2	-	-

Amov...

AmovPercMrun...

Name	Usage	Description
AmovPercMevaSlowWpstDf	-	-
AmovPercMevaSlowWpstDfl	-	-
AmovPercMevaSlowWpstDfr	-	-
AmovPercMevaSlowWrflDf	p	Runs forward a few steps
AmovPercMevaSlowWrflDfl	-	-
AmovPercMevaSlowWrflDfr	-	-
AmovPercMevaSnonWnonDf	-	-
AmovPercMevaSnonWnonDfl	-	-
AmovPercMevaSnonWnonDfr	-	-
AmovPercMrunSlowWlnrDb	-	-
AmovPercMrunSlowWlnrDbl	-	-
AmovPercMrunSlowWlnrDbr	-	-
AmovPercMrunSlowWlnrDf	-	-
AmovPercMrunSlowWlnrDfl	-	-
AmovPercMrunSlowWlnrDfr	-	-

AmovPercMrunSlowWlnrDI	-	-
AmovPercMrunSlowWlnrDnon_transition	-	-
AmovPercMrunSlowWlnrDr	-	-
AmovPercMrunSlowWpstDb	-	-
AmovPercMrunSlowWpstDbI	-	-
AmovPercMrunSlowWpstDbr	-	-
AmovPercMrunSlowWpstDf	-	-
AmovPercMrunSlowWpstDf_AmovPercMstpSrasWpstDnon_gthArm	-	-
AmovPercMrunSlowWpstDf_AmovPercMstpSrasWpstDnon_gthEnd	-	-
AmovPercMrunSlowWpstDf_AmovPercMstpSrasWpstDnon_gthStart	p	Throws a grenade
AmovPercMrunSlowWpstDf_AmovPercMstpSrasWpstDnon_gthThrow	-	-
AmovPercMrunSlowWpstDfI	-	-
AmovPercMrunSlowWpstDfr	-	-
AmovPercMrunSlowWpstDI	-	-
AmovPercMrunSlowWpstDnon_transition	-	-
AmovPercMrunSlowWpstDr	-	-
AmovPercMrunSlowWrfIDb	-	-
AmovPercMrunSlowWrfIDbI	-	-
AmovPercMrunSlowWrfIDbr	-	-
AmovPercMrunSlowWrfIDf	-	-
AmovPercMrunSlowWrfIDf_AmovPercMevaSrasWrfIDb	-	-
AmovPercMrunSlowWrfIDf_AmovPercMevaSrasWrfIDI	-	-
AmovPercMrunSlowWrfIDf_AmovPercMevaSrasWrfIDr	-	-
AmovPercMrunSlowWrfIDf_AmovPercMstpSrasWrfIDnon_gthEnd	-	-
AmovPercMrunSlowWrfIDf_AmovPercMstpSrasWrfIDnon_gthStart	-	-
AmovPercMrunSlowWrfIDf_AmovPercMstpSrasWrfIDnon_gthThrow	-	-
AmovPercMrunSlowWrfIDf_AmovPpneMstpSrasWrfIDnon	-	-
AmovPercMrunSlowWrfIDfI	-	-
AmovPercMrunSlowWrfIDfr	-	-
AmovPercMrunSlowWrfIDI	-	-
AmovPercMrunSlowWrfIDnon_transition	p	Runs a few steps, kneels down, returns
AmovPercMrunSlowWrfIDr	-	-
AmovPercMrunSnonWnonDb	-	-
AmovPercMrunSnonWnonDbI	-	-
AmovPercMrunSnonWnonDbr	-	-
AmovPercMrunSnonWnonDf	-	-
AmovPercMrunSnonWnonDf_AmovPercMstpSnonWnonDnon_gthEnd	-	-
AmovPercMrunSnonWnonDf_AmovPercMstpSnonWnonDnon_gthStart	-	-
AmovPercMrunSnonWnonDf_AmovPercMstpSnonWnonDnon_gthThrow	-	-
AmovPercMrunSnonWnonDfI	-	-
AmovPercMrunSnonWnonDfr	-	-
AmovPercMrunSnonWnonDI	-	-
AmovPercMrunSnonWnonDr	-	-
AmovPercMrunSrasWrfIDb	-	-
AmovPercMrunSrasWrfIDbI	-	-
AmovPercMrunSrasWrfIDbr	-	-
AmovPercMrunSrasWrfIDf	-	-
AmovPercMrunSrasWrfIDf_AmovPercMevaSrasWrfIDb	-	-
AmovPercMrunSrasWrfIDf_AmovPercMevaSrasWrfIDI	-	-
AmovPercMrunSrasWrfIDf_AmovPercMevaSrasWrfIDr	-	-
AmovPercMrunSrasWrfIDfI	-	-
AmovPercMrunSrasWrfIDfr	-	-
AmovPercMrunSrasWrfIDI	-	-
AmovPercMrunSrasWrfIDr	-	-

AmovPercMspr...	
Name	Usage Description
AmovPercMsprSlowWpstDf	- -
AmovPercMsprSlowWpstDf_AmovPpneMstpSrasWpstDnon	- -
AmovPercMsprSlowWpstDf_AmovPpneMstpSrasWpstDnon_2	- -
AmovPercMsprSlowWpstDfl	- -
AmovPercMsprSlowWpstDfr	- -
AmovPercMsprSlowWrflDf	- -
AmovPercMsprSlowWrflDf_AllSprint	- -
AmovPercMsprSlowWrflDf_AmovPpneMstpSrasWrflDnon	- -
AmovPercMsprSlowWrflDf_AmovPpneMstpSrasWrflDnon_2	- -
AmovPercMsprSlowWrflDfl	- -
AmovPercMsprSlowWrflDfl_AllSprint	- -
AmovPercMsprSlowWrflDfr	- -
AmovPercMsprSlowWrflDfr_AllSprint	- -
AmovPercMsprSnonWnonDf	- -
AmovPercMsprSnonWnonDf_AmovPpneMstpSnonWnonDnon	- -
AmovPercMsprSnonWnonDf_AmovPpneMstpSnonWnonDnon_2	- -
AmovPercMsprSnonWnonDfl	- -
AmovPercMsprSnonWnonDfr	- -
AmovPercMsprSrasWrflDf	- -
AmovPercMsprSrasWrflDf_AllSprint	- -
AmovPercMsprSrasWrflDfl	- -
AmovPercMsprSrasWrflDfl_AllSprint	- -
AmovPercMsprSrasWrflDfr	- -
AmovPercMsprSrasWrflDfr_AllSprint	- -
AmovPercMstp...	
Name	Usage Description
AmovPercMstpSlowWrflDnon	- -
AmovPercMstpSlowWrflDnon_AmovPercMevaSrasWrflDb	- -
AmovPercMstpSlowWrflDnon_AmovPercMevaSrasWrflDl	- -
AmovPercMstpSlowWrflDnon_AmovPercMevaSrasWrflDr	- -
AmovPercMstpSlowWrflDnon_AmovPercMstpSrasWrflDnon	- -
AmovPercMstpSlowWrflDnon_AmovPpsitMstpSlowWrflDnon	p Sits on ground
AmovPercMstpSlowWrflDnon_Salute	- -
AmovPercMstpSlowWrflDnon_SaluteIn	p Salutes
AmovPercMstpSlowWrflDnon_SaluteOut	- -
AmovPercMstpSlowWrflDnon_seeWatch	p Checks watch with weapon in other hand
AmovPercMstpSlowWrflDnon_talking	- -
AmovPercMstpSlowWrflDnon_turnL	- -
AmovPercMstpSlowWrflDnon_turnR	- -
AmovPercMstpSnonWnonDnon	- -
AmovPercMstpSnonWnonDnon_AcrgPknMstpSnonWnonDnon_getInHigh	- -
AmovPercMstpSnonWnonDnon_AcrgPknMstpSnonWnonDnon_getInLow	- -
AmovPercMstpSnonWnonDnon_AcrgPknMstpSnonWnonDnon_getInMedium	- -
AmovPercMstpSnonWnonDnon_AinvPknMstpSnonWnonDnon	p kneels down (inventory check)
AmovPercMstpSnonWnonDnon_AmovPercMstpSrasWpstDnon	p Makes two steps back, draws his pistol and moves some steps forward
AmovPercMstpSnonWnonDnon_AmovPercMstpSrasWpstDnon_end	p Draws pistol and moves

		some steps forward
AmovPercMstpSnonWnonDnon_AmovPercMstpSrasWrflDnon	p	Takes weapon from back and return it to the back
AmovPercMstpSnonWnonDnon_AmovPercMstpSsurWnonDnon	p	Puts hands behind his head and back
AmovPercMstpSnonWnonDnon_AmovPknlMstpSnonWnonDnon	p	Kneels with one knee down
AmovPercMstpSnonWnonDnon_AmovPknlMstpSrasWlnrDnon	-	-
AmovPercMstpSnonWnonDnon_AmovPpneMstpSnonWnonDnon	p	Lays down, then stands up
AmovPercMstpSnonWnonDnon_AmovPsitMstpSnonWnonDnon_ground	p	Sits on the ground
AmovPercMstpSnonWnonDnon_AwopPercMstpSoptWbinDnon	p	Stands; switches to binocular, looks and put it back.
AmovPercMstpSnonWnonDnon_AwopPercMstpSoptWbinDnon_end	-	-
AmovPercMstpSnonWnonDnon_Ease	p	"At ease"
AmovPercMstpSnonWnonDnon_EaseIn	-	-
AmovPercMstpSnonWnonDnon_EaseOut	-	-
AmovPercMstpSnonWnonDnon_Salute	-	-
AmovPercMstpSnonWnonDnon_SaluteIn	-	-
AmovPercMstpSnonWnonDnon_SaluteOut	-	-
AmovPercMstpSnonWnonDnon_carCheckPush	-	-
AmovPercMstpSnonWnonDnon_carCheckWash	-	-
AmovPercMstpSnonWnonDnon_carCheckWheel	-	-
AmovPercMstpSnonWnonDnon_exerciseKata	p	Martial arts moves
AmovPercMstpSnonWnonDnon_exercisePushup	p	Pushups
AmovPercMstpSnonWnonDnon_exercisekneeBendA	p	Situps
AmovPercMstpSnonWnonDnon_exercisekneeBendB	-	-
AmovPercMstpSnonWnonDnon_normalizationTest	-	-
AmovPercMstpSnonWnonDnon_seeWatch	-	-
AmovPercMstpSnonWnonDnon_talking	p	Having a conversions (lips not moving)
AmovPercMstpSnonWnonDnon_turnL	-	-
AmovPercMstpSnonWnonDnon_turnR	-	-
AmovPercMstpSrasWpstDnon	-	-
AmovPercMstpSrasWpstDnon_AinvPknlMstpSnonWnonDnon	p	Kneels in front of ammo box.
AmovPercMstpSrasWpstDnon_AinvPknlMstpSnonWnonDnon_end	-	-
AmovPercMstpSrasWpstDnon_AmovPercMstpSnonWnonDnon	-	-
AmovPercMstpSrasWpstDnon_AmovPercMstpSnonWnonDnon_end	-	-
AmovPercMstpSrasWpstDnon_AmovPercMstpSrasWrflDnon	-	-
AmovPercMstpSrasWpstDnon_AmovPercMstpSrasWrflDnon_end	-	-
AmovPercMstpSrasWpstDnon_AmovPknlMstpSrasWpstDnon	-	-
AmovPercMstpSrasWpstDnon_AmovPpneMstpSrasWpstDnon	-	-
AmovPercMstpSrasWpstDnon_AwopPercMstpSoptWbinDnon	-	-
AmovPercMstpSrasWpstDnon_AwopPercMstpSoptWbinDnon_end	-	-
AmovPercMstpSrasWpstDnon_AwopPercMstpSoptWbinDnon_mid	-	-
AmovPercMstpSrasWpstDnon_Salute	-	-
AmovPercMstpSrasWpstDnon_SaluteIn	-	-
AmovPercMstpSrasWpstDnon_SaluteIn_end	-	-
AmovPercMstpSrasWpstDnon_SaluteOut	-	-
AmovPercMstpSrasWpstDnon_SaluteOut_end	-	-
AmovPercMstpSrasWpstDnon_turnL	-	-
AmovPercMstpSrasWpstDnon_turnR	-	-

AmovPercMstpSrasWrflDnon	p	Kneels in front of ammo box. Stays that way
AmovPercMstpSrasWrflDnon_AinvPknlMstpSlayWrflDnon	p	(same?)
AmovPercMstpSrasWrflDnon_AmovPercMevaSrasWrflDb	-	-
AmovPercMstpSrasWrflDnon_AmovPercMevaSrasWrflDl	-	-
AmovPercMstpSrasWrflDnon_AmovPercMevaSrasWrflDr	-	-
AmovPercMstpSrasWrflDnon_AmovPercMstpSlowWrflDnon	-	-
AmovPercMstpSrasWrflDnon_AmovPercMstpSnonWnonDnon	-	-
AmovPercMstpSrasWrflDnon_AmovPercMstpSrasWpstDnon	p	switch to pistol. draws a pistol
AmovPercMstpSrasWrflDnon_AmovPercMstpSrasWpstDnon_end	-	-
AmovPercMstpSrasWrflDnon_AmovPknlMstpSrasWrflDnon	-	-
AmovPercMstpSrasWrflDnon_AmovPpneMstpSrasWrflDnon	-	-
AmovPercMstpSrasWrflDnon_AwopPercMstpSoptWbinDnon	-	-
AmovPercMstpSrasWrflDnon_AwopPercMstpSoptWbinDnon_end	-	-
AmovPercMstpSrasWrflDnon_turnL	-	-
AmovPercMstpSrasWrflDnon_turnR	-	-
AmovPercMstpSsurWnonDnon	-	-
AmovPercMstpSsurWnonDnon_AmovPercMstpSnonWnonDnon	-	-
AmovPercMwlk...		
Name	Usage	Description
AmovPercMwlkSlowWrflDb	-	-
AmovPercMwlkSlowWrflDbl	-	-
AmovPercMwlkSlowWrflDbr	-	-
AmovPercMwlkSlowWrflDf	p	Walks a few steps straight
AmovPercMwlkSlowWrflDfl	p	Walks a few frontsidesteps left
AmovPercMwlkSlowWrflDfr	p	Walks a few frontsidesteps left
AmovPercMwlkSlowWrflDl	p	Walks a few backsidesteps left."
AmovPercMwlkSlowWrflDr	-	-
AmovPercMwlkSnonWnonDb	-	-
AmovPercMwlkSnonWnonDbl	-	-
AmovPercMwlkSnonWnonDbr	-	-
AmovPercMwlkSnonWnonDf	p	Puts a weapon on back, walks a bit, takes weapon on hand again
AmovPercMwlkSnonWnonDfl	-	-
AmovPercMwlkSnonWnonDfr	-	-
AmovPercMwlkSnonWnonDl	-	-
AmovPercMwlkSnonWnonDr	-	-
AmovPercMwlkSrasWpstDb	-	-
AmovPercMwlkSrasWpstDbl	-	-
AmovPercMwlkSrasWpstDbr	-	-
AmovPercMwlkSrasWpstDf	-	-
AmovPercMwlkSrasWpstDf_AmovPercMstpSrasWpstDnon_gthArm	-	-
AmovPercMwlkSrasWpstDf_AmovPercMstpSrasWpstDnon_gthEnd	-	-
AmovPercMwlkSrasWpstDf_AmovPercMstpSrasWpstDnon_gthStart	-	-
AmovPercMwlkSrasWpstDf_AmovPercMstpSrasWpstDnon_gthThrow	-	-
AmovPercMwlkSrasWpstDf_AwopPercMrunSgthWnonDf_1	-	-
AmovPercMwlkSrasWpstDfl	-	-
AmovPercMwlkSrasWpstDfr	-	-
AmovPercMwlkSrasWpstDl	-	-
AmovPercMwlkSrasWpstDr	-	-
AmovPercMwlkSrasWrflDb	-	-
AmovPercMwlkSrasWrflDbl	-	-
AmovPercMwlkSrasWrflDbr	-	-

AmovPercMwlkSrasWrflDf	-	-
AmovPercMwlkSrasWrflDf_AmovPercMstpSrasWrflDnon_gthEnd	-	-
AmovPercMwlkSrasWrflDf_AmovPercMstpSrasWrflDnon_gthStart	-	-
AmovPercMwlkSrasWrflDf_AmovPercMstpSrasWrflDnon_gthThrow	-	-
AmovPercMwlkSrasWrflDfl	-	-
AmovPercMwlkSrasWrflDfr	-	-
AmovPercMwlkSrasWrflDI	-	-
AmovPercMwlkSrasWrflDnon_transition	-	-
AmovPercMwlkSrasWrflDr	-	-

AmovPknIMrun...

Name	Usage	Description
AmovPknIMrunSlowWpstDb	-	-
AmovPknIMrunSlowWpstDbl	-	-
AmovPknIMrunSlowWpstDbr	-	-
AmovPknIMrunSlowWpstDf	-	-
AmovPknIMrunSlowWpstDfl	-	-
AmovPknIMrunSlowWpstDfr	-	-
AmovPknIMrunSlowWpstDI	-	-
AmovPknIMrunSlowWpstDr	-	-
AmovPknIMrunSlowWrflDb	-	-
AmovPknIMrunSlowWrflDbl	-	-
AmovPknIMrunSlowWrflDbr	-	-
AmovPknIMrunSlowWrflDf	p	Kneels, walks few steps, kneels, runs a few steps
AmovPknIMrunSlowWrflDfl	-	-
AmovPknIMrunSlowWrflDfr	-	-
AmovPknIMrunSlowWrflDI	-	-
AmovPknIMrunSlowWrflDr	-	-
AmovPknIMrunSnonWnonDb	-	-
AmovPknIMrunSnonWnonDbl	-	-
AmovPknIMrunSnonWnonDbr	-	-
AmovPknIMrunSnonWnonDf	-	-
AmovPknIMrunSnonWnonDfl	-	-
AmovPknIMrunSnonWnonDfr	-	-
AmovPknIMrunSnonWnonDI	-	-
AmovPknIMrunSnonWnonDr	-	-
AmovPknIMrunSrasWrflDb	-	-
AmovPknIMrunSrasWrflDbl	-	-
AmovPknIMrunSrasWrflDbr	-	-
AmovPknIMrunSrasWrflDf	-	-
AmovPknIMrunSrasWrflDfl	-	-
AmovPknIMrunSrasWrflDfr	-	-
AmovPknIMrunSrasWrflDI	-	-
AmovPknIMrunSrasWrflDr	-	-

AmovPknIMspr...

Name	Usage	Description
AmovPknIMsprSlowWpstDf	-	-
AmovPknIMsprSlowWpstDfl	-	-
AmovPknIMsprSlowWpstDfr	-	-
AmovPknIMsprSlowWrflDf	-	-
AmovPknIMsprSlowWrflDf_AllSprint	-	-
AmovPknIMsprSlowWrflDfl	-	-
AmovPknIMsprSlowWrflDfl_AllSprint	-	-
AmovPknIMsprSlowWrflDfr	-	-

```

AmovPknLMsprSlowWrflDfr_AllSprint - -
AmovPknLMsprSnonWnonDf - -
AmovPknLMsprSnonWnonDfl - -
AmovPknLMsprSnonWnonDfr - -
AmovPknLMsprSrasWrflDf - -
AmovPknLMsprSrasWrflDf_AllSprint - -
AmovPknLMsprSrasWrflDfl - -
AmovPknLMsprSrasWrflDfl_AllSprint - -
AmovPknLMsprSrasWrflDfr - -
AmovPknLMsprSrasWrflDfr_AllSprint - -

```

AmovPknLMstp...

Name	Usage	Description
AmovPknLMstpSlowWrflDnon	p	Runs a few steps, returns.
AmovPknLMstpSlowWrflDnon_turnL	-	-
AmovPknLMstpSlowWrflDnon_turnR	-	-
AmovPknLMstpSnonWnonDnon	-	-
AmovPknLMstpSnonWnonDnon_AinvPknLMstpSnonWnonDnon	-	-
AmovPknLMstpSnonWnonDnon_AmovPercMsprSnonWnonDf	-	-
AmovPknLMstpSnonWnonDnon_AmovPercMsprSnonWnonDf_2	-	-
AmovPknLMstpSnonWnonDnon_AmovPercMstpSnonWnonDnon	-	-
AmovPknLMstpSnonWnonDnon_AmovPpneMstpSnonWnonDnon	-	-
AmovPknLMstpSnonWnonDnon_turnL	-	-
AmovPknLMstpSnonWnonDnon_turnR	-	-
AmovPknLMstpSrasWlnrDnon	-	-
AmovPknLMstpSrasWlnrDnon_AmovPercMstpSnonWnonDnon	-	-
AmovPknLMstpSrasWlnrDnon_AmovPknLMstpSrasWpstDnon	-	-
AmovPknLMstpSrasWlnrDnon_AmovPknLMstpSrasWpstDnon_end	-	-
AmovPknLMstpSrasWlnrDnon_AmovPknLMstpSrasWrflDnon	-	-
AmovPknLMstpSrasWlnrDnon_AmovPpneMstpSnonWnonDnon	-	-
AmovPknLMstpSrasWlnrDnon_AwopPknLMstpSoptWbinDnon	-	-
AmovPknLMstpSrasWlnrDnon_AwopPknLMstpSoptWbinDnon_End	-	-
AmovPknLMstpSrasWlnrDnon_turnL	-	-
AmovPknLMstpSrasWlnrDnon_turnR	-	-
AmovPknLMstpSrasWpstDnon	-	-
AmovPknLMstpSrasWpstDnon_AinvPknLMstpSnonWnonDnon	p	Kneels in front of ammo box.
AmovPknLMstpSrasWpstDnon_AinvPknLMstpSnonWnonDnon_end	-	-
AmovPknLMstpSrasWpstDnon_AmovPercMsprSrasWpstDf	-	-
AmovPknLMstpSrasWpstDnon_AmovPercMsprSrasWpstDf_2	-	-
AmovPknLMstpSrasWpstDnon_AmovPercMstpSrasWpstDnon	-	-
AmovPknLMstpSrasWpstDnon_AmovPknLMstpSrasWlnrDnon	-	-
AmovPknLMstpSrasWpstDnon_AmovPknLMstpSrasWlnrDnon_end	-	-
AmovPknLMstpSrasWpstDnon_AmovPknLMstpSrasWrflDnon	-	-
AmovPknLMstpSrasWpstDnon_AmovPknLMstpSrasWrflDnon_end	-	-
AmovPknLMstpSrasWpstDnon_AmovPpneMstpSrasWpstDnon	-	-
AmovPknLMstpSrasWpstDnon_AwopPknLMstpSoptWbinDnon	-	-
AmovPknLMstpSrasWpstDnon_AwopPknLMstpSoptWbinDnon_end	-	-
AmovPknLMstpSrasWpstDnon_AwopPknLMstpSoptWbinDnon_mid	-	-
AmovPknLMstpSrasWpstDnon_turnL	-	-
AmovPknLMstpSrasWpstDnon_turnR	-	-
AmovPknLMstpSrasWrflDnon	-	-
AmovPknLMstpSrasWrflDnon_AinvPknLMstpSlayWrflDnon	p	Kneels in front of ammo box, then returns to normal kneeling position
AmovPknLMstpSrasWrflDnon_AmovPercMstpSrasWrflDnon	-	-
AmovPknLMstpSrasWrflDnon_AmovPknMevaSrasWrflDb	-	-

AmovPknIMstpSrasWrflDnon_AmovPknIMevaSrasWrflDI	-	-
AmovPknIMstpSrasWrflDnon_AmovPknIMevaSrasWrflDr	-	-
AmovPknIMstpSrasWrflDnon_AmovPknIMsprSrasWrflDf	-	-
AmovPknIMstpSrasWrflDnon_AmovPknIMsprSrasWrflDf_2	-	-
AmovPknIMstpSrasWrflDnon_AmovPknIMstpSrasWlnrDnon	-	-
AmovPknIMstpSrasWrflDnon_AmovPknIMstpSrasWpstDnon	-	-
AmovPknIMstpSrasWrflDnon_AmovPknIMstpSrasWpstDnon_end	-	-
AmovPknIMstpSrasWrflDnon_AmovPpneMstpSrasWrflDnon	-	-
AmovPknIMstpSrasWrflDnon_AwopPknIMstpSoptWbinDnon	-	-
AmovPknIMstpSrasWrflDnon_AwopPknIMstpSoptWbinDnon_end	-	-
AmovPknIMstpSrasWrflDnon_turnL	-	-
AmovPknIMstpSrasWrflDnon_turnR	-	-

AmovPknIMwlk...

Name	Usage	Description
AmovPknIMwlkSlowWrflDb	-	-
AmovPknIMwlkSlowWrflDbI	-	-
AmovPknIMwlkSlowWrflDbr	-	-
AmovPknIMwlkSlowWrflDf	p	Walks a few step crouched
AmovPknIMwlkSlowWrflDfI	-	-
AmovPknIMwlkSlowWrflDfr	-	-
AmovPknIMwlkSlowWrflDI	-	-
AmovPknIMwlkSlowWrflDr	-	-
AmovPknIMwlkSnonWnonDb	-	-
AmovPknIMwlkSnonWnonDbI	-	-
AmovPknIMwlkSnonWnonDbr	-	-
AmovPknIMwlkSnonWnonDf	-	-
AmovPknIMwlkSnonWnonDfI	-	-
AmovPknIMwlkSnonWnonDfr	-	-
AmovPknIMwlkSnonWnonDI	-	-
AmovPknIMwlkSnonWnonDr	-	-
AmovPknIMwlkSrasWlnrDb	-	-
AmovPknIMwlkSrasWlnrDbI	-	-
AmovPknIMwlkSrasWlnrDbr	-	-
AmovPknIMwlkSrasWlnrDf	-	-
AmovPknIMwlkSrasWlnrDfI	-	-
AmovPknIMwlkSrasWlnrDfr	-	-
AmovPknIMwlkSrasWlnrDI	-	-
AmovPknIMwlkSrasWlnrDr	-	-
AmovPknIMwlkSrasWpstDb	-	-
AmovPknIMwlkSrasWpstDbI	-	-
AmovPknIMwlkSrasWpstDbr	-	-
AmovPknIMwlkSrasWpstDf	-	-
AmovPknIMwlkSrasWpstDfI	-	-
AmovPknIMwlkSrasWpstDfr	-	-
AmovPknIMwlkSrasWpstDI	-	-
AmovPknIMwlkSrasWpstDr	-	-
AmovPknIMwlkSrasWrflDb	-	-
AmovPknIMwlkSrasWrflDbI	-	-
AmovPknIMwlkSrasWrflDbr	-	-
AmovPknIMwlkSrasWrflDf	-	-
AmovPknIMwlkSrasWrflDfI	-	-
AmovPknIMwlkSrasWrflDfr	-	-
AmovPknIMwlkSrasWrflDI	-	-
AmovPknIMwlkSrasWrflDnon_transition	-	-

AmovPknIMwIkSrasWrflDr - -

AmovPpneMrun...

Name	Usage	Description
------	-------	-------------

AmovPpneMrunSlowWpstDb	-	-
AmovPpneMrunSlowWpstDbl	-	-
AmovPpneMrunSlowWpstDbr	-	-
AmovPpneMrunSlowWpstDf	-	-
AmovPpneMrunSlowWpstDfl	-	-
AmovPpneMrunSlowWpstDfr	-	-
AmovPpneMrunSlowWpstDI	-	-
AmovPpneMrunSlowWpstDr	-	-
AmovPpneMrunSlowWrflDb	-	-
AmovPpneMrunSlowWrflDbl	-	-
AmovPpneMrunSlowWrflDbr	-	-
AmovPpneMrunSlowWrflDf	-	-
AmovPpneMrunSlowWrflDfl	-	-
AmovPpneMrunSlowWrflDfr	-	-
AmovPpneMrunSlowWrflDI	-	-
AmovPpneMrunSlowWrflDr	-	-
AmovPpneMrunSnonWnonDb	-	-
AmovPpneMrunSnonWnonDbl	-	-
AmovPpneMrunSnonWnonDbr	-	-
AmovPpneMrunSnonWnonDf	p	puts weapon on back, goes to ground, takes weapon back, does something
AmovPpneMrunSnonWnonDfl	-	-
AmovPpneMrunSnonWnonDfr	-	-
AmovPpneMrunSnonWnonDI	-	-
AmovPpneMrunSnonWnonDr	-	-

AmovPpneMspr...

Name	Usage	Description
------	-------	-------------

AmovPpneMsprSlowWrflDf	-	-
AmovPpneMsprSlowWrflDfl	-	-
AmovPpneMsprSlowWrflDfr	-	-
AmovPpneMsprSnonWnonDf	-	-
AmovPpneMsprSnonWnonDfl	-	-
AmovPpneMsprSnonWnonDfr	-	-

AmovPpneMstp...

Name	Usage	Description
------	-------	-------------

AmovPpneMstpSnonWnonDnon	p	goes prone - reloads, gets up
AmovPpneMstpSnonWnonDnon_AmovPercMsprSnonWnonDf	p	prone - gets up, runs forward, takes prim. weapon
AmovPpneMstpSnonWnonDnon_AmovPercMsprSnonWnonDf_2	p	prone - gets up, runs forward, takes prim. weapon
AmovPpneMstpSnonWnonDnon_AmovPercMstpSnonWnonDnon	p	prone - gets up, takes prim. weapon
AmovPpneMstpSnonWnonDnon_AmovPknIMstpSnonWnonDnon	p	prone - kneels, goes prone - takes prim. weapon, gets up
AmovPpneMstpSnonWnonDnon_AmovPknIMstpSrasWlnrDnon	p	prone - kneels, takes sec. weapon, stands up
AmovPpneMstpSnonWnonDnon_AmovPpneMevaSnonWnonDI	p	prone - rolls to left, takes prim. weapon, gets up
AmovPpneMstpSnonWnonDnon_AmovPpneMevaSnonWnonDr	p	prone - rolls to right, takes

		prim. weapon, gets up
AmovPpneMstpSnonWnonDnon_AmovPpneMstpSrasWpstDnon	p	prone - takes handgun
AmovPpneMstpSnonWnonDnon_AmovPpneMstpSrasWpstDnon_end	p	prone - takes handgun
AmovPpneMstpSnonWnonDnon_AmovPpneMstpSrasWrflDnon	p	prone - gets up, takes prim. weapon
AmovPpneMstpSnonWnonDnon_AmovPsitMstpSnonWnonDnon_ground	p	prone - sits up, stands up, takes prim. weapon
AmovPpneMstpSnonWnonDnon_AwopPpneMstpSoptWbinDnon	p	prone - takes binoculars, puts them away, takes prim. weapon, gets up
AmovPpneMstpSnonWnonDnon_AwopPpneMstpSoptWbinDnon_end	p	prone - takes binoculars, puts them away, takes prim. weapon, gets up
AmovPpneMstpSnonWnonDnon_healed	p	prone - moves head, takes prim. weapon, gets up
AmovPpneMstpSnonWnonDnon_turnL	p	prone - takes prim. weapon, gets up
AmovPpneMstpSnonWnonDnon_turnR	p	prone - takes prim. weapon, gets up
AmovPpneMstpSrasWpstDnon	p	prone - aiming handgun
AmovPpneMstpSrasWpstDnon_AmovPercMsprSlowWpstDf	p	prone - runs forward, holsters handgun, takes prim. weapon
AmovPpneMstpSrasWpstDnon_AmovPercMsprSlowWpstDf_2	p	crouched - runs forward, holsters handgun, takes prim. weapon
AmovPpneMstpSrasWpstDnon_AmovPercMstpSrasWpstDnon	p	prone - gets up, holsters handgun, takes prim. weapon
AmovPpneMstpSrasWpstDnon_AmovPknIMstpSrasWpstDnon	p	prone - kneels, aims handgun
AmovPpneMstpSrasWpstDnon_AmovPpneMevaSlowWpstDI	p	prone - rolls to left, aims handgun
AmovPpneMstpSrasWpstDnon_AmovPpneMevaSlowWpstDr	p	prone - rolls to right, aims handgun
AmovPpneMstpSrasWpstDnon_AmovPpneMstpSnonWnonDnon	p	prone - holsters handgun, takes prim. weapon, gets up
AmovPpneMstpSrasWpstDnon_AmovPpneMstpSnonWnonDnon_end	p	prone - holsters handgun, takes prim. weapon, gets up
AmovPpneMstpSrasWpstDnon_AmovPpneMstpSrasWrflDnon	p	prone - holsters handgun, takes prim. weapon, gets up
AmovPpneMstpSrasWpstDnon_AmovPpneMstpSrasWrflDnon_end	p	prone - holsters handgun, takes prim. weapon, gets up
AmovPpneMstpSrasWpstDnon_AwopPpneMstpSoptWbinDnon	p	prone - holsters handgun, takes binoculars, puts them away, takes handgun
AmovPpneMstpSrasWpstDnon_AwopPpneMstpSoptWbinDnon_end	p	prone - takes binoculars, puts them away, takes handgun
AmovPpneMstpSrasWpstDnon_AwopPpneMstpSoptWbinDnon_mid	p	prone - holsters handgun, takes binoculars, puts them away, takes handgun
AmovPpneMstpSrasWpstDnon_healed	p	prone - moves head around
AmovPpneMstpSrasWpstDnon_turnL	p	prone - aims handgun
AmovPpneMstpSrasWpstDnon_turnR	p	prone - aims handgun
AmovPpneMstpSrasWrflDnon	p	prone - gets up
AmovPpneMstpSrasWrflDnon_AmovPercMsprSlowWrflDf	p	prone - gets up, runs forward
AmovPpneMstpSrasWrflDnon_AmovPercMsprSlowWrflDf_2	p	crouching - gets up, runs forward
AmovPpneMstpSrasWrflDnon_AmovPercMstpSrasWrflDnon	p	prone - gets up

AmovPpneMstpSrasWrflDnon_AmovPknIMstpSrasWrflDnon	p	prone - kneels, gets up
AmovPpneMstpSrasWrflDnon_AmovPpneMevaSlowWrflDI	p	prone - aims handgun, rolls to left, gets up
AmovPpneMstpSrasWrflDnon_AmovPpneMevaSlowWrflDr	p	prone - aims handgun, rolls to right, gets up
AmovPpneMstpSrasWrflDnon_AmovPpneMstpSnonWnonDnon	p	prone - holsters prim. weapon, takes it back, gets up
AmovPpneMstpSrasWrflDnon_AmovPpneMstpSrasWpstDnon	p	prone - holsters prim. weapon, takes handgun
AmovPpneMstpSrasWrflDnon_AmovPpneMstpSrasWpstDnon_end	p	prone - takes handgun, aims
AmovPpneMstpSrasWrflDnon_AwopPpneMstpSoptWbinDnon	p	prone - takes binoculars, gets up, puts them away, takes prim. weapon
AmovPpneMstpSrasWrflDnon_AwopPpneMstpSoptWbinDnon_end	p	prone - takes binoculars, gets up, uses them again, puts them away, takes prim. weapon
AmovPpneMstpSrasWrflDnon_healed	p	prone - moves head, gets up
AmovPpneMstpSrasWrflDnon_turnL	p	prone - gets up
AmovPpneMstpSrasWrflDnon_turnR	p	prone - gets up

AmovPsitMstp...

Name	Usage	Description
AmovPsitMstpSlowWrflDnon	-	-
AmovPsitMstpSlowWrflDnon_AmovPercMstpSlowWrflDnon	p	Gets up from sitting
AmovPsitMstpSlowWrflDnon_Smoking	p	Sitting and smoking
AmovPsitMstpSlowWrflDnon_WeaponCheck1	p	Sitting and checking weapon
AmovPsitMstpSlowWrflDnon_WeaponCheck2	-	-
AmovPsitMstpSnonWnonDnon_AmovPercMstpSnonWnonDnon_ground	-	-
AmovPsitMstpSnonWnonDnon_ground	p	Gets up from sitting on ground
AmovPsitMstpSnonWnonDnon_ground_AmovPpneMstpSnonWnonDnon	p	Sits, goes prone, rises back up

Aswm...

Name	Usage	Description
AswmPercMrunSnonWnonDf	-	-
AswmPercMrunSnonWnonDf_AswmPercMstpSnonWnonDnon	p	Swimming
AswmPercMsprSnonWnonDf	-	-
AswmPercMstpSnonWnonDnon	-	-
AswmPercMstpSnonWnonDnon_AswmPercMrunSnonWnonDf	-	-
AswmPercMwlcSnonWnonDf	-	-

Awop

Name	Usage	Description
AwopPercMstpSgthWnonDnon_end	-	-
AwopPercMstpSgthWnonDnon_start	-	-
AwopPercMstpSgthWnonDnon_throw	-	-
AwopPercMstpSgthWpstDnon_Part1	p	Throws a grenade
AwopPercMstpSgthWpstDnon_Part2	-	-
AwopPercMstpSgthWpstDnon_Part3	-	-
AwopPercMstpSgthWpstDnon_Part4	-	-
AwopPercMstpSgthWpstDnon_Part5	-	-
AwopPercMstpSgthWrflDnon_End1	-	-
AwopPercMstpSgthWrflDnon_End2	-	-
AwopPercMstpSgthWrflDnon_Start1	p	Throws a grenade
AwopPercMstpSgthWrflDnon_Start2	-	-

AwopPercMstpSgthWrflDnon_Throw1	-	-
AwopPercMstpSgthWrflDnon_Throw2	-	-
AwopPercMstpSoptWbinDnon_AmovPercMstpSnonWnonDnon	-	-
AwopPercMstpSoptWbinDnon_AmovPercMstpSnonWnonDnon_end	-	-
AwopPercMstpSoptWbinDnon_AmovPercMstpSrasWpstDnon	-	-
AwopPercMstpSoptWbinDnon_AmovPercMstpSrasWpstDnon_end	-	-
AwopPercMstpSoptWbinDnon_AmovPercMstpSrasWpstDnon_mid	-	-
AwopPercMstpSoptWbinDnon_AmovPercMstpSrasWrflDnon	-	-
AwopPercMstpSoptWbinDnon_AmovPercMstpSrasWrflDnon_end	-	-
AwopPercMstpSoptWbinDnon_AwopPknLMstpSoptWbinDnon_pst	-	-
AwopPercMstpSoptWbinDnon_AwopPknLMstpSoptWbinDnon_rifle	-	-
AwopPercMstpSoptWbinDnon_AwopPpneMstpSoptWbinDnon_non	-	-
AwopPercMstpSoptWbinDnon_AwopPpneMstpSoptWbinDnon_pst	-	-
AwopPercMstpSoptWbinDnon_AwopPpneMstpSoptWbinDnon_rifle	-	-
AwopPercMstpSoptWbinDnon_non	-	-
AwopPercMstpSoptWbinDnon_pst	-	-
AwopPercMstpSoptWbinDnon_rfl	p	Uses binoculars
AwopPknLMstpSgthWpstDnon_Part1	-	-
AwopPknLMstpSgthWpstDnon_Part2	-	-
AwopPknLMstpSgthWpstDnon_Part3	-	-
AwopPknLMstpSgthWpstDnon_Part4	-	-
AwopPknLMstpSgthWpstDnon_Part5	-	-
AwopPknLMstpSgthWrflDnon_End	-	-
AwopPknLMstpSgthWrflDnon_Start	p	Kneeling; throws a grenade
AwopPknLMstpSgthWrflDnon_Throw	-	-
AwopPknLMstpSoptWbinDnon_AmovPknLMstpSrasWlnrDnon	-	-
AwopPknLMstpSoptWbinDnon_AmovPknLMstpSrasWlnrDnon_end	-	-
AwopPknLMstpSoptWbinDnon_AmovPknLMstpSrasWpstDnon	-	-
AwopPknLMstpSoptWbinDnon_AmovPknLMstpSrasWpstDnon_end	-	-
AwopPknLMstpSoptWbinDnon_AmovPknLMstpSrasWpstDnon_mid	-	-
AwopPknLMstpSoptWbinDnon_AmovPknLMstpSrasWrflDnon	-	-
AwopPknLMstpSoptWbinDnon_AmovPknLMstpSrasWrflDnon_end	-	-
AwopPknLMstpSoptWbinDnon_AwopPercMstpSoptWbinDnon_pst	-	-
AwopPknLMstpSoptWbinDnon_AwopPercMstpSoptWbinDnon_rifle	-	-
AwopPknLMstpSoptWbinDnon_AwopPpneMstpSoptWbinDnon_pst	-	-
AwopPknLMstpSoptWbinDnon_AwopPpneMstpSoptWbinDnon_rifle	-	-
AwopPknLMstpSoptWbinDnon_lnr	-	-
AwopPknLMstpSoptWbinDnon_pst	-	-
AwopPknLMstpSoptWbinDnon_rfl	-	-
AwopPpneMstpSgthWnonDnon_end	-	-
AwopPpneMstpSgthWnonDnon_start	-	-
AwopPpneMstpSgthWnonDnon_throw	p	Throws a grenade
AwopPpneMstpSgthWpstDnon_Part1	-	-
AwopPpneMstpSgthWpstDnon_Part2	-	-
AwopPpneMstpSgthWpstDnon_Part3	-	-
AwopPpneMstpSgthWpstDnon_Part4	-	-
AwopPpneMstpSgthWpstDnon_Part5	-	-
AwopPpneMstpSgthWrflDnon_End	-	-
AwopPpneMstpSgthWrflDnon_Start	-	-
AwopPpneMstpSgthWrflDnon_Throw	-	-
AwopPpneMstpSoptWbinDnon_AmovPpneMstpSnonWnonDnon	-	-
AwopPpneMstpSoptWbinDnon_AmovPpneMstpSnonWnonDnon_end	-	-
AwopPpneMstpSoptWbinDnon_AmovPpneMstpSrasWpstDnon	-	-
AwopPpneMstpSoptWbinDnon_AmovPpneMstpSrasWpstDnon_end	-	-

AwopPpneMstpSoptWbinDnon_AmovPpneMstpSrasWpstDnon_mid	-	-
AwopPpneMstpSoptWbinDnon_AmovPpneMstpSrasWrflDnon	-	-
AwopPpneMstpSoptWbinDnon_AmovPpneMstpSrasWrflDnon_end	-	-
AwopPpneMstpSoptWbinDnon_AwopPercMstpSoptWbinDnon_non	-	-
AwopPpneMstpSoptWbinDnon_AwopPercMstpSoptWbinDnon_pst	-	-
AwopPpneMstpSoptWbinDnon_AwopPercMstpSoptWbinDnon_rifle	-	-
AwopPpneMstpSoptWbinDnon_AwopPknMstpSoptWbinDnon_pst	-	-
AwopPpneMstpSoptWbinDnon_AwopPknMstpSoptWbinDnon_rifle	-	-
AwopPpneMstpSoptWbinDnon_non	-	-
AwopPpneMstpSoptWbinDnon_pst	-	-
AwopPpneMstpSoptWbinDnon_rfl	-	-

Basic...

Name	Usage	Description
BasicDriver	s	Sitting (low)
BasicDriverDead	-	
BasicDriverDying	s	Sitting (high) [same as crew?]
BasicDriverOut	-	-
BasicDriverOutDead	-	-
BasicDriverOutDying	-	-
BasicSittingGunner	-	-
BasicSittingGunner_Dead	-	-

Ladder...

Name	Usage	Description
LadderBase	X	WILL CRASH ARMA!
LadderDownBase	X	WILL CRASH ARMA!
LadderDownEnd	p	steps off the bottom of a ladder
LadderDownLoop	p	endlessly climbs down a ladder
LadderDownStart	p	starts at the top of a ladder
LadderStatic	p	stays still on a ladder
LadderUpBase	X	WILL CRASH ARMA!
LadderUpEnd	p	steps off the top of a ladder
LadderUpLoop	p	endlessly climbs up ladder
LadderUpStart	p	starts at the bottom of a ladder

Misc

Name	Usage	Description
Crew	s	Sitting (high)
DeadState	s	Dies
LauncherReloadKneel	-	-
para_pilot	s	parachuting pose
PistolMagazineReloadKneel	-	-
PistolMagazineReloadProne	-	-
PistolMagazineReloadStand	-	-
SprintBaseDf	s	runs in one direction endlessly
SprintBaseDfl	s	runs in circle endlessly
SprintBaseDfr	-	-
SprintCivilBaseDf	-	-
SprintCivilBaseDfl	-	-
SprintCivilBaseDfr	-	-
TestDance	-	-
TestFlipflop	-	-
TestJabbaFun	-	-
TestROM	-	-

TestROMFingers	-	-
TestSurrender	-	-
TransAnimBase	-	-
WeaponMagazineReloadKneel	-	-
WeaponMagazineReloadProne	p	takes weapon off shoulder, goes prone, reloads, gets up again
WeaponMagazineReloadStand	-	-

Weitergehend möchte ich den interessierten Leser davon in Kenntnis setzen, dass bereits an weiteren Versionen gearbeitet wird und die englische Version bereits Alphastatus erreicht hat. Nach dem Beispiel von BIS missbrauche ich die deutsche Fangemeinschaft als Tester. Weitere Versionen werden nach Kontrolle und Ideensammlung folgen.

Anregungen, Wünsche, Fehler bitte mit kurzer Erklärung in unserem Forum

<http://deadeye.pytalhost.com/manual/index.php>

hinterlassen, oder an mich Alex.Sworn@lycos.de schicken oder mich über ICQ 344 005 013 benachrichtigen.

Danke

Quellen:

Animationsliste:

http://community.bistudio.com/wiki/ArmA:_Moves

Script Commands:

http://community.bistudio.com/wiki/Category:Scripting_Commands_ArMA

Waffenliste:

http://community.bistudio.com/wiki/ArmA:_Weapons

und das geballte gesammelte Wissen der Clanmitglieder der Luftlandebrigade 31.

Zu den Rechten.

Von uns für euch. Das heißt macht hiermit was ihr wollt. Druckt es, bearbeitet es weiter oder verbrennt es. Mir egal. Es steht euch alles frei.