

AMS Camscripting Tutorial

"Einstieg und Erläuterung des Camscripting in ArmA"

Autor: Imutep
Version: 1.0
Stand: 10. April 2008
Website: http://www.assaultmissionstudio.de

Inhaltsverzeichnis:

Kapitel 1: Die Kamera/Steuerung 03
1.1 Steuerung der Kamera 03
1.2 Kamera starten 04

Kapitel 2: Position der Kamera 05
2.1 Der Koordinatenblock 06
2.2 Die X-Y-Z Koordinaten 07
2.3 Objekte selektieren 08

Kapitel 3: Die erste Szene 09
3.1 Anfang des Szenen-Scripts 09
3.2 Ende des Szenen-Scripts 10

Kapitel 4: Die Kamerafahrt 11
4.1 Kordinatenblöcke einer Kamerafahrt 12
4.2 Fertiges Script einer Kamerafahrt 13

Kapitel 5: Erweiterte Kamerafahrt 14
5.1 Script der erweiterten Kamerafahrt 15
5.2 Weiche Kamerafahrt 16
5.3 Kameraposition vorberechnen lassen 16

Kapitel 6: Kamera verfolgt Heli/Fahrzeug 17
6.2 Szenenwechsel bei Verfolgung 20
6.3 Erweiteter Szenenwechsel bei Verfolgung 21

Kapitel 7: Kamera an Heli/Fahrzeug .....WIP..... 00

Kapitel 8: Text und Musik einbinden 22
8.1 Text-und Einblendeffekte 22
8.2 Musik und Sound einbinden 24
8.3 Sounds/Funktexte ausschalten 26
8.4 Kamera im Nachtsicht und Fernglasmodus 26

Danksagung/Credits 26

Begrüssung:

Zunächst einmal vielen Dank, dass du dich für unser Tutorial interessierst.

Wir, das Team vom **Assault Mission Studio** sind bakanntlich begeisterte Mapper und haben in den glorreichen Zeiten von Operation Flashpoint unsere ersten Erfahrungen im Bereich Editing gemacht. Und wir können sagen, dass es eine schwierige Zeit war, bis wir unsere ersten guten Ergebnisse erzielt haben.

Unser sogenanntes Gesellenstück im Mapping war zweifellos die BW Afghan Kampagne **Unexpected War,** an der wir fast 2 Jahre arbeiteten. Schon mit der LibMod-Kampagne **Ostfront 1941-45** von Stoned Boy, wurde uns klar, wie viel Spass man eigentlich beim editieren haben kann, wenn man seine ersten Erfolge eingeheimst hat. Doch dafür braucht es Zeit, Ausdauer und natürlich das Erlernen des Camscripting.

Um auch euch den Einstieg ins Camscripting einwenig zu erleichtern, haben wir uns entschlossen, dieses Tutorial zu schreiben, in der Hoffnung, dass es dem ein oder anderen ein wichtige Hillfestellung darbietet.

Vorwort:

Wie schon in Operation Flashpoint, bietet uns auch ArmA die Möglichkeit, unsere Missionen anhand von Videosequenzen etwas aufzuwerten. Wer kennt nicht aus den Missionen die schönen Einleitungen und zum Teil wilden Kamerafahrten? Ja, das ist schon fast Pflichtprogramm bei Mappern geworden.

Die Arbeit an einer Sequenz erfordert die meiste Arbeit an einer Mission, bringt also genügend Zeit mit, wenn ihr ein Intro oder eine Zwischensequenz erstellen wollt. Gut Ding braucht seine Weile! ;-)

Ihr denkt bestimmt, das kriege ich nie hin! Das kapier ich nie? Quatsch sag ich da nur. Wenn ihr dieses Tutorial intensiv abgearbeitet habt, sind solche Sequenzen ein „Kinderspiel" für euch und ihr werdet wesentlich mehr Freude am editieren haben, wenn ihr auch das Camscripten einigermasen beherrscht.

In diesem Tutorial werden wir gezielt auf die Möglichkeiten von ArmA selbst eingehen, ohne gross auf die endlosen Scriptmöglichkeiten, welche ausserhalb von ArmA geschrieben werden einzugehen. Denn über das Thema Camscripting könnte man ein ganzes Buch schreiben, wenn man noch alle Scriptmöglichkeiten erläutern würde.

Alle Beispiel haben wir ausführlich erklärt und mit vielen Beispielbildern illustriert, um dieses Tutorial zu einem brauchbaren Nachschlagewerk für Einsteiger darzustellen.
Auf unserer **AMS**-Homepage unter **http://www.assaultmissionstudio.de** findet ihr einige Scripte, die ihr in eure Szenen einbauen könnt um so noch interessantere Sequenzen zu erstellen.

Ich werde auch einige Beispiel erläutern, wie ihr die Kamera an ein fahrendes Fahrzeug oder Flugzeug anhängt, um so eine Verfolgungs-Szene zu erstellen. Die Möglichkeiten der Texteinblendung, sowie das Einbinden von Musikstücken in eine Sequenz werde ich in diesem Tutorial ebenfalls erläutern.

Diese Tutorial wird euch längst nicht zu Profis im Bereich Camscripting machen, jedoch soll es in erster Linie dazu dienen, euch den Einstieg, einige Kenntnisse und den Umgang mit der Kamera in ArmA lehren. Ihr werdet am Ende fähig sein, Intros, Outros und Zwischensequenzen für eure Mission erstellen zu können.

Zubehör:

Was benötige ich alles um eine Szene zu erstellen?

- ArmA Editor
- Notepad oder Script Editor
- entsprechender Missionsordner
- Ideen, etwas Geduld und vor allem Zeit

Sehr zu empfehlen ist der **ArmA Script Editor** von Chris Henderman, welchen ihr ebenfalls auf unserer Seite unter den Tools und Hilfsprogrammen findet.

Doch genug der Worte, lasst uns loslegen mit dem ersten Kapitel!

Kapitel 1: Die Kamera und ihre Steuerung

In diesem Kapitel werde ich euch die Kamera und ihre Steuerung erklären. Die Steuerung der Kamera ist recht einfach und kann gemütlich über die Tastatur geführt werden.

Anhand der Steuerung könnt ihr im Kameramodus auf der Karte entlang „schweben" und die Kamera in die optimale Position für eine Szene bringen. Bevor wir nun die Kamera starten, widmen wir uns zunächst der Steuerung.

Die Steuerung:

Hier nun die wichtigsten Tastenbelegungen:

W oder **Pfeiltaste hoch**	Kamera forwärts bewegen
S oder **Pfeiltaste runter**	Kamera rückwärts bewegen
A oder **Pfeiltaste links**	Kamera nach links bewegen
D oder **Pfeiltaste rechts**	Kamera nach rechts bewegen
L	Fadenkreuz an–oder ausschalten
V	Kamera ausschalten
F oder **Strg-Taste**	Objekt oder Einheit selektieren
Q oder **Bild auf**	Kamera nach oben bewegen
Y oder **Bild ab**	Kamera nach unten bewegen
8 (Numpad)	Kamera schaut aufwärts
2 (Numpad)	Kamera schaut abwärts
4 (Numpad)	Kamera schwenken(nach links drehen
6 (Numpad)	Kamera schwenken/nach rechts drehen
+ (Numpad)	Kamera ranzoomen
- (Numpad)	Kamera wegzoomen
Shift-Taste	Kamera beschleunigen
Maus vor/zurück	Kamera vor-und zurückbewegen
Linke Maustaste	Kameraposition speichern

Im ersten Moment möge euch die Steuerung etwas kompliziert erscheinen, doch mit etwas Übung habt ihr die Steuerung der Kamera schnell im Griff. Ihr könnt so ganz bequem mit beiden Händen an der Tastatur die Kamera steuern und in die entsprechende Position bringen.

Im Laufe dieses Tutorials werdet ihr den Sinn und Nutzen der einzelnen Tasten erlernen, deswegen werde ich jetzt nicht näher darauf eingehen.

Wenn ihr euch nun die Steuerung etwas angesehen habt, können wir uns nun der Kamera widmen.

Hinweis:
Die Steuerung ist abhängig von eurer jeweiligen Einstellung der Tastenbelegung. Hier habe ich die bei mir funktionierende Tastenbelegung aufgelistet.

Kamera starten:

Startet nun den Editor, stellt ihn auf „Einleitung" und platziert eine Einheit (Spieler) auf der Karte. In der Init-Zeile dieser Einheit schreiben wir den entsprechenden Script-Befehl rein, der die Kamera aktiviert. Ihr könnt natürlich diesen Befehl auch in ein beliebiges Fahrzeug oder Objekt schreiben. In unserem Beispiel nutze ich eine BLUFOR-Einheit (Spieler). Der Befehl um die Kamera zu erstellen lautet:

this exec „camera.sqs";	ODER	Name exec „camera.sqs";



Wenn ihr jetzt im Editor auf Vorschau klickt, befindet ihr euch nun im Kameramodus.



Mit dem Befehl **showCinemBorder false;** befindet ihr euch im Vollbild-Modus, ohne die beiden schwarzen Streifen ober-und unterhalb der Vorschau. Nun könnt ihr auf der Karte rumschweben und die Steuerung etwas testen und euch damit vertraut machen, bevor wir uns an die erste Szene wagen.

Hinweis:
Die Blickrichtung der Kamera zu Beginn ist immer richtung Norden, von der Sicht der Karte im Editor aus gesehen.

Kapitel 2: Die erste Position der Kamera

Nachdem wir nun wissen wie wir die Kamera erstellen und uns etwas mit der Steuerung vertraut gemacht haben, ist es an der Zeit die erste Position der Kamera festzulegen.
Doch bevor wir nun ans eingemachte gehen, müssen wir noch unser Notepad öffnen, oder den ArmA Script Editor starten und ein neues Script-File öffnen. Mit den Tasten **Strg, Alt** und **Entf** kommt ihr zurück zu eurem Desktop. Startet den Script Editor und öffnet ein neues Script File. Speichert dieses nun als **scene.sqs** in eurem entsprechendem Missionsordner ab. Natürlich könnt ihr statt „scene" das Script auch anders benennen. Wenn ihr das neue Script File geöffnet und gespeichert habt, geht ihr wieder ins Spiel zurück.

Da wir ja schon eine Einheit auf der Karte platziert haben, wollen wir nun ein wenig mehr Leben in unsere erste Szene reinbringen. Setzt nun einige Meter vor der Einheit ein beliebiges Fahrzeug. Anschliessend geht ihr wieder in den Kameramodus zurück und steuert die Kamera, bis ihr euer Fahrzeug im Blick habt. Wenn ihr der Meinung seid, ihr habt eine gute Position gefunden, so werden wir diese nun als unsere erste Szene nehmen.



Abbildung 1: Die erste Szene im Kameramodus

Drückt nun die linke Maustaste, jetzt werden die Kamerakoordinaten im Zwischenspeicher abgelegt.
Mit **Strg, Alt und Entf** geht ihr nun wieder zurück zum Desktop und öffnet euer „scene.sqs"-Script.
Mit **Strg+V** fügt ihr nun die gespeicherten Kamerakoordinaten in euer Script ein. Das geht auch indem ihr die rechte Maustaste drückt und dann auf einfügen klickt.
Wenn ihr also alles richtig gemacht habt, müsste das ganze dann so aussehen.

```
;=== 22:52:36
_camera camPrepareTarget [73059.49,73765.57,18.13]
_camera camPreparePos [2686.51,2716.40,2.01]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
```

Abbildung 2: Die Szene aus dem Kameramodus als Datenblock

Prima! Eure erste Position der Kamera ist im Kasten.
Bevor wir uns an die nächste Szene machen, sehen wir uns den Datenblock etwas genauer an.

Hinweis:
Mit der Taste „L" könnt ihr das nervige Fadenkreuz abstellen. Das benötigen wir erst später im Verlauf.

Der Datenblock:

Ihr habt nun eure erste Szene als Datenblock in der scene.sqs abgespeichert, somit widmen wir uns nun dem Datenblock. Er enthält alle Daten einer Szene und kann auch entsprechen bearbeitet werden, um beispielsweise den Zoom zu ändern, oder eine Kamerafahrt zu erstellen.

```
;=== 22:52:36
_camera camPrepareTarget [73059.49,73765.57,18.13]
_camera camPreparePos [2686.51,2716.40,2.01]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
```

Abbildung 3: Der Datenblock

Folgende Daten sind enthalten:

;=== 22:52:36
Die Zeit, wann die Kameraposition erstellt wurde. Statt der Uhrzeit könnt ihr auch Infos zur Position eingeben um die Übersicht zu behalten. Bsp: ***;=== 22:52:36—Blick Stryker***
_camera camPrepareTarget [73059.49,73765.57,18.13]
Die Blickrichtung der Kamera. Hier ist die Kamera auf eine Position zentriert. Man kann auch Namen von Objekten oder Einheiten angeben. Doch dazu später mehr.
_camera camPreparePos [2686.51,2716.40,2.01]
Die Position der Kamera (X,Y,Z Koordinaten der Insel) X=Waagrechter Wert; Y=Senkrechter Wert; Z=Höhenwert;
_camera camPrepareFOV 0.700
Der Zoom der Kamera. Je kleiner der angegebene Wert, je grösser wird der Zoom
_camera camCommitPrepared 0
Die Zeit, wie lange die Kamera zu dieser Position braucht. 0 steht für sofort da. Je grösser der Wert, umso länger geht die Kamerafahrt.
@camCommitted _camera
Hier pausiert das Script, bis die Kamera ihre Position erreicht hat. Erst dann liest das Script die nächste Szene aus.

Auf den ersten Blich wirkt so ein Datenblock etwas kompliziert, aber da wir im Verlauf dieses Tutorials eine komplette Sequenz erstellen, werdet ihr sicherlich auch den Umgang mit diesen Daten einigermasen beherrschen.

Hinweis:
Wenn ihr statt der Uhrzeit (**;=== 22:52:36**) Infos eingeben wollt, achtet darauf, dass ihr davor das Semikolon (;) stehen lasst. Was nach dem Semikolon steht, wird vom Script nicht gelesen. Das ist wichtig, da sonst ein Fehler im Script entsteht.

Die X-Y-Z Koordinaten:

Die X-Y-Z Koordinaten geben an, wie die Kamera zum Ziel ausgerichtet ist. Hier nun eine kurze Erklärung anhand der **_camera camPreparePos [2686.51,2716.40,2.01].** Ich habe zur besseren Verständlichkeit die einzelnen Werte der Koordinaten farblich hervorgehoben.

Die hier vorgegebenen Koordinaten wurden automatisch von ArmA ermittelt, anhand der Speicherung einer Position im Kameramodus. Da wir auch ohne Kameramodus die Position der Kamera festlegen können, werde ich nun die Möglichkeiten und den Umgang mit diesen Werten etwas genauer erläutern.

Der **X-Wert** gibt uns die waagrechten Koordinaten an.

Der **Y-Wert** gibt uns die senkrechten Koordinaten an.

Der **Z-Wert** gibt uns die Höhe der Kamera zum Ziel an.

Bestimmtes Objekt selektieren:

Es ist auch möglich, an der Position der Kamera das Fahrzeug direkt anzuselektieren. Das hat den Effekt, dass wenn das Fahrzeug fährt, die Kamera auf ihrer Position bleibt und dabei das Fahrzeug ins Visier nimmt. Die Kamera schaut also dem Fahrzeug hinterher, bleibt aber auf ihrer Position. Das ganze probieren wir nun anhand unseres Beispiels. Hierfür sollten wir jetzt unserem Fahrzeug einen Namen geben. Da ich den Stryker nutze, nenn ich ihn einfach mal **Stryker** ;-)

Geht nun wieder in den Kameramodus und nehmt euer Fahrzeug ins Visier. Lasst aber das Fadenkreuz jetzt an, denn das brauchen wir jetzt.
Bewegt die Kamera so, dass das Fadenkreuz auf euer Fahrzeug zeigt. Anschliessend drückt ihr die Taste „F" oder „Strg". Jetzt müsste die Kamera auf euer Fahrzeug fixiert sein und ein rotes Viereck um das Fadekreuz ist entstanden. Ist das Viereck gelb, so ist das Fahrzeug nicht selektiert. Wenn ihr alles richtig gemacht habt, müsst das in etwa so aussehen.



Abbildung 4: Selektiertes Fahrzeug im Kameramodus

Jetzt habt ihr die Möglichkeit den Blickwinkel und Position der Kamera zu ändern, während die Kamera weiterhin das Fahrzeug anvisiert. Bringt nun die Kamera in eure gewünschte Position. Drückt anschliessend wieder die linke Maustaste und speichert die Position wieder als Datenblock in eurem Script ab. Ihr könnt den vorhin gespeicherten Datenblock überschreiben.

```
;=== 1:02:19
_camera camPrepareTarget Stryker
_camera camPreparePos [2689.56,2715.08,5.44]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
```

Abbildung 5: Datenblock des selektiertem Fahrzeugs

Was hat sich hier verändert? Richtig, unter **_camera camPrepareTarget** sind nun keine Koordinaten angegeben, sondern der Name des anselektierten Fahrzeugs. In diesem Beispiel also der Stryker. Ebenso habe ich die Position der Kamera verändert.

Hinweis:
Wenn ihr die Kamera bewegt, während ihr das Fahrzeug anselektiert, nicht die Nummerntasten (Numpads) nutzen, da sonst das Fadenkreuz abgeschalten wird. Also die Kamera nur nach links, rechts, rauf und runter bewegen.

Kapitel 3: Die erste Szene

Nachdem wir uns nun ausgiebig dem Datenblock auseinandergesetzt haben, wollen wir nun an unserer ersten Szene weiterarbeiten. Widmen wir uns wieder unserer ersten Position des Fahrzeugs. Wir haben die Position der Kamera nun in unserem Script abgespeichert, können also wieder zurück in den Editor gehen.
Jetzt bringen wir etwas Fahrt in unsere Sequenz! Hierfür geben wir unserem Stryker Wegpunkte vor und schicken ihn so auf eine Fahrt. Wohin ist völlig egal, hauptsache er fährt ;-)
Um die Kamerasequenz nachher auch beenden zu können, benötigen wir noch einen Auslöser als Typ **Ende1** mit der Bedingung z.B **ende1**. Das bewirkt, dass die Einleitung durch die Bedinguung **ende1** beendet wird und später die Mission auch starten kann.
Habt ihr die Wegmarke(n) und den Ende1-Auslöser platziert, so speichert eure Mission (Einleitung) ab.

Jetzt widmen wir uns wieder unserem Script, der scene.sqs. Wir haben unsere Position der Kamera, nun müssen wir die Kamera aber noch erstellen, bevor wir uns die Szene anschauen können. Hierfür müssen wir unsere scene.sqs mit weiteren Daten erweitern und vervollständigen, das machen wir wie folgt.

```
titlecut [" ","BLACK IN",5]
_camera = "camera" camcreate [0,0,0]
_camera cameraeffect ["internal", "back"]


;=== 1:02:19
_camera camPrepareTarget Stryker
_camera camPreparePos [2689.56,2715.08,5.44]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera

~10

Player cameraEffect ["terminate","back"]
camDestroy _camera
ende1=true;
Exit;
```

Abbildung 6: Erste fertige Scene im Kamera-Script

Wie ihr erkennt, hat sich unser Script um einige Daten erweitert.
Am Anfang unseres Scriptes habe ich die Kamera erstellt. Am Ende die Kamera und das Script beendet. Unsere erste kleine Kamerasequenz ist geboren. Sehen wir uns die neuen Daten nun etwas genauer an.

Anfang des Scriptes:

titlecut [" ","BLACK IN",5]
Hier wird von schwarzem Bildschirm aus in die Szene eingeblendet. Hier habe ich die Dauer von 5 Sekunden eingegeben.
_camera = "camera" camcreate [0,0,0]
*Die Kamera wird an einer bestimmten Position erstellt (X,Y,Z). Zu Beginn der Sequenz sind die X,Y,Z-Koordinaten auf 0 gestellt. Später werden wir hier die Daten aus einer **_camera camPreparePos** eingeben.*
_camera cameraeffect ["internal", "back"]
Der Effekt der Kamera. Das „internal" ist die Standard-Einstellung.

Hinweis:
Durch drücken der linken Maustaste im Kameramodus könnt ihr nur immer eine Position je Klick abspeichern. Für jede weitere Position müsst ihr erneut in den Kameramodus.

Ende des Scriptes:

~10
Zeitangabe in Sekunden, bis die nächste Scene startet oder das Script endet. In unserem Script wird also nach einer Dauer von 10 Sekunden das Script wieder beendet.
Player cameraEffect ["terminate","back"]
Gibt die Kamera an den Spieler zurück und beginnt die Kamera zu beenden.
camDestroy _camera
Hier wird die Kamera „zerstört"- also abgeschalten.
ende1=true;
Die Bedingung des Typ Ende1 Auslösers wird auf „true" gestellt. Somit ist die Szene beendet.
Exit;
Das Script wird beendet.

Juhu!! *Luftschlange* :-)
Nun haben wir also unsere erste kleine Kamerasequenz erstellt, welche wir uns natürlich sofort im Editor genüsslich anschauen werden. Nachher werden wir Schritt für Schritt unser Camscript immer mehr erweitern und ausbauen.

Wir haben unser Script **scene.sqs** abgespeichert? Gut, dann gehen wir nun wieder zurück in unseren Editor und rufen unsere Einleitung auf.

Damit wir nun endlich in den Genuss unserer ersten kleinen Szene kommen, müssen wir natürlich noch das entsprechende Script aufrufen lassen. Das machen wir, indem wir in der Initalisierungszeile der Spieler-Einheit folgenden Befehl eingeben.

```
This exec „scene.sqs“;
```

Anstatt „scene" müsst ihr natürlich den Namen eures Scriptes eingeben, falls ihr es nicht so benannt habt. Speichert anschliessend erneut die Mission (Einleitung) ab und drückt auf „Vorschau".

Ihr könnt natürlich das Script auch über einen Auslöser startetn lassen, doch das würde ich nicht in einer Einleitung nutzen, da man zu Beginn der Sequenz sonst für 1 Sekunde die Spielereinheit sieht und das wirkt nicht schön. Für eine Sequenz während der Mission eignet sich ein Auslöser hervorragend.

Wenn ihr alles richtig gemacht habt, startet eure Szene mit Blick auf das Fahrzeug, es fährt los und die Kamera schaut für 10 Sekunden hinterher, ehe sie beendet wird und euch von der Sequenz in den Editor zurückwirft.

Is das der Fall, so ist alles richtig abgelaufen und wir können mit dem nächsten Kapitel beginnen.

Hinweis:
Experimentiert einwenig mit **titlecut [" ","BLACK IN",5], _camera camPrepareFOV 0.700** und der Zeitverzögerung **~10** um auch mal andere Ergebnisse zu erzielen. Denn Übung macht den Meister ;-)

Kapitel 4: Die Kamerafahrt

Nachdem wir nun endlich unsere erste kleine Szene im Kasten haben, wollen wir nun unsere Sequenz etwas erweitern, in Form einer kleinen Kamerafahrt.

Was ist eine Kamerafahrt? Nun, hier bewegt sich die Kamera von Position A zu Position B in Dauer einer bestimmten Zeit und kann zum Beispiel während sie sich zur nächsten Position begibt, einem fahrenden Fahrzeug hinterherblicken. Das ist ganz leicht. Lasst uns also nun wieder in den Editor zu unserem Stryker zurückkehren. Positioniert ihn irgendwo auf der Karte und gebt ihm einige Wegmarken vor.

In unserer ersten Szene, haben wir lediglich eine Kameraposition genutzt, jetzt werden wir unser Script um eine weitere Position erweitern. Für eine Kamerafahrt benötigen wir logischwerweise mindestens 2 Kamerapositionen.

Erstellt nun eure erste Position, speichert sie wieder in eurem Script ab und erstellt anschliessend die zweite Position. Speichert diese ebenfalls wieder in eurem Script ab (Linke Maustaste --->Str+V einfügen ins Script)
Ihr könnt diese Kamerafahrt auch mit der Position unserer ersten Szene und einer neuen Position machen. ;-)



Abbildung 7: Erste Kameraposition

Für unser Beispiel habe ich eine Position gewählt, die den Stryker seitlich oben anvisiert, die zweite Position wählte ich hinter dem Stryker unten.



Abbildung 8: Zweite Kameraposition

Der Datenblock zu unseren beiden Positionen:

Nachdem ihr nun die beiden Positionen der Kamera als Datenblöcke in unserem Script abgespeichert haben, widmen wir uns nun den beiden Datenblöcken, welche wir nun etwas bearbeiten müssen, um eine Kamerafahrt zu erstellen.

```
;=== 18:33:44--obere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2824.00,2917.12,10.27]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera

;=== 18:32:49--untere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2839.51,2917.06,1.10]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
```

Abbildung 9: Die beiden Kamerapositionen als Datenblöcke

Hier haben wir nun unsere 2 Datenblöcke, welche die beiden Positionen darstellen. Für dieses Beispiel habe ich nun eine obere Position zum Stryker und eine unter Position.
Wenn ihr jetzt von beiden Datenblöcken die Koordinaten aus der **_camera camPreparePos** vergleicht, so erkennt ihr an dem Z-Wert, dass sich die Höhe zum ausgerichteten Stryker von 10.27m auf 1.10m verringert hat. Unsere Kamerafahrt geht demnach von der oberen zur unteren Position. Ihr könnt die beiden Blöcke auch vertauschen wenn ihr wollt, dann fährt die Kamera von der unteren zur oberen Position. Wie ihr wollt. ;-)

Doch noch haben wir keine Kamerafahrt. Es fehlen uns noch kleinere Erweiterungen des Scriptes und des Datenblocks der unteren Position.

```
;=== 18:33:44--obere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2824.00,2917.12,10.27]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera

~3

;=== 18:32:49--untere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2839.51,2917.06,1.10]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 7
@camCommitted _camera
```

Abbildung 10: Die beiden Datenblöcke für Kamerafahrt erweitert

Was haben wir geändert? Richtig, zwischen den beiden Datenblöcken haben wir nun wieder einen Zeitverzögerungswert in Sekunden eingegeben (**~3**). Das heisst, dass die Kamera für 3 Sekunden an der oberen Position verharrt und dabei auf den Stryker fixiert ist, ehe sie zur unteren Position weiterfährt und dort ebenfalls den Stryker anvisiert.

Doch für die Fahrt müssen wir nun die **_camera camCommitPrepared** der unteren Position bearbeiten. Ich habe hier also den Wert **0** durch den Wert **7** ersetzt. Das hat den Effekt, dass die Kamera nun 7 Sekunden braucht um die untere Position anzufahren. Jetzt haben wir unsere kleine Kamerafahrt erstellt. Sehen wir uns nun das fertige Script unserer Kamerafahrt (scene.sqs) etwas genauer an.

Das fertige Script zur Kamerafahrt:

Bevor wir uns nun das fertige Script zu unserer Kamerafahrt ansehen, gehen wir nochmal kurz die Änderungen und Erweiterungen unserer beiden Datenblöcke durch.

Zunächst haben wir zwischen die beiden Datenblöcke einen Zeitverzögerungswert in Sekunden (**~3**) eingetragen. Dieser Wert bestimmt die sogenannte Wartezeit, bis die Kamera sich zur nächsten Position begibt. Ihr könnt hier z.B. auch 5 oder 10 eintragen. Dann würde die Kamera 5 oder 10 Sekunden warten.
Und schlieslich haben wir noch die **_camera camCommitPrepared** der unteren Position bearbeitet. Der hier eingetragene Wert bestimmt die „Fahrzeit" in Sekunden, wie lange die Kamera brauchen soll, um zur nächsten Position zu fahren. Wir haben hier den Wert **0** mit dem Wert **7** ersetzt. Demnach fährt die Kamera nun mit einer Dauer von 7 Sekunden zur nächsten Position. Würden wir den Wert auf 0 belassen, so würde die Kamera nicht fahren, sondern direkt an die Position gesetzt. Auch dieser Wert ist variabel und ihr könnt die Fahrt auch 10 Sekunden dauern lassen.

Sehen wir uns nun das fertige Script an:

```
titlecut [" ","BLACK IN",5]
_camera = "camera" camcreate [0,0,0]
_camera cameraeffect ["internal", "back"]


;=== 18:33:44--obere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2824.00,2917.12,10.27]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera

~3

;=== 18:32:49--untere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2839.51,2917.06,1.10]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 7
@camCommitted _camera

~3

titlecut [" ","BLACK Out",5]
~1
Player cameraEffect ["terminate","back"]
camDestroy _camera
ende1=true;
Exit;
```

Abbildung 11: Fertige Kamerafahrt als Script

Wie ihr seht, ist unser Script nun ziemlich gewachsen.
Am Ende des Scriptes habe ich noch **titlecut [" ","BLACK Out",5]** eingetragen. Hier wird die Kamera schwarz ausgeblendet, mit einer Dauer von 5 Sekunden. Somit wird unsere Sequenz nicht sofort beendet, sondern schön mit einem Ausblendeffdekt. Wenn ihr auf die erste Zeile im Script schaut, so haben wir die Kamera schwarz einblenden lassen. Ebenfalls mit einer Dauer von 5 Sekunden.

Der Zeitverzögerungswert (**~1**) nach **titlecut [" ","BLACK Out",5]** bewirkt, dass die Kamera noch 1 Sekunde schwarz bleibt, ehe sie beendet wird. Wählt hier einen Wert zwischen 1 und 2 Sekunden, für ein optimales Ergebnis.
Speichert nun euer Script ab und schaut es euch im Editor an. Unsere erste Kamerafahrt ist fertig!

Kapitel 5: Erweiterte Kamerafahrt mit Szenenwechsel

Wir haben nun unsere erste kleine Kamerafahrt im Kasten. Jetzt wollen wir noch einen Schritt weiter gehen und nach unserer kleinen Kamerafahrt einen Szenenwechsel einbinden. Mit Szenenwechsel meine ich, dass die Kamera, nachdem sie den Stryker für einige Zeit nachgeschaut hat, zu einer anderen Position wechselt, ehe sie beendet wird. Das ist ganz einfach ;-)

Platziert nun zusätzlich ein beliebiges Fahrzeug irgendwo auf der Karte und einige Meter weiter weg einen **Soldaten**. Dem Soldat gebt ihr den Namen „Soldat" und gebt ihm Wegmarken vor, so dass er irgendwo hingeht. Ich habe für unser Beispiel noch weitere Einheiten ringsum postiert, um das ganze etwas lebendiger zu gestalten.

Geht nun in den Kameramodus und speichert eine Position vor dem Fahrzeug unten und eine weitere oberhalb des Fahrzeugs und selektiert von dort den Soldaten mit dem Namen „Soldat". Wie in meinem Beispiel.



Abbildung 12: Position Fahrzeug unten



Abbildung 13: Position Fahrzeug oben mit Blick auf selektierten Soldaten

Nun haben wir 2 neue Kamera-Datenblöcke, welche wir nun in unsere scene.sqs einfügen. Und zwar vor dem Befehl **titlecut** **[" ","BLACK Out",5]** in unserem Script.

Nun sollte unser Script folgendermasen aussehen:

```
titlecut [" ","BLACK IN",5]
_camera = "camera" camcreate [0,0,0]
_camera cameraeffect ["internal", "back"]


;=== 18:33:44--obere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2824.00,2917.12,10.27]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera

~3

;=== 18:32:49--untere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2839.51,2917.06,1.10]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 7
@camCommitted _camera

~3

;=== 20:06:27--Fahrzeug unten
_camera camPrepareTarget [16134.26,-96230.73,15.64]
_camera camPreparePos [2879.56,2886.94,0.99]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera

~3

;=== 20:13:42---Fahrzeug oben mit Blick auf Soldat
_camera camPrepareTarget Soldat
_camera camPreparePos [2884.51,2886.41,5.07]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 6
@camCommitted _camera

~10

titlecut [" ","BLACK Out",5]
~1
Player cameraEffect ["terminate","back"]
camDestroy _camera
ende1=true;
Exit;
```

Wenn ihr nun das Script genauer betrachtet, werdet ihr merken, dass wir unsere beiden neuen Datenblöcke nach dem letzten Blick auf den Stryker eingefügt haben.
Wir haben also nachdem letzten Blick auf den Stryker eine Zeitverzögerung von **3** Sekunden, danach wird die Kamera ohne Fahrt zur neuen Position springen, wo wir unser Fahrzeug platziert haben. Ich wählte deshalb ein Fahrzeug, um euch zu zeigen, dass man es auch ohne zu selektieren anvisieren kann.

Nachdem unsere Kamera für **3** Sekunden auf ihrer Position das Fahrzeug im Blick hat, wechselt die Kamera zur nächsten Position, oberhalb des Fahrzeugs und wird dabei unseren Soldaten anvisieren. Die Kamerafahrt zu dieser Position dauert **6** Sekunden. Hier wird nun die Kamera unseren Soldaten für **10** Sekunden nachschauen, ehe sie schwarz ausgeblendet wird und die Szene endet. Im Grunde ist hier nichts neues. **Lest bitte erst weiter, wenn ihr es bis hierher verstanden habt. Sonst wird's zu kompliziert!!**

Die weiche Kamerafahrt

Nachdem wir ja nun unsere erste Kamerafahrt erstellt haben, ist es an der Zeit, mal eine weiche Kamerafahrt zu erstellen. Unter weiche Kamerafahrt versteht man, dass die Kamera nicht durch den Befehl **@camCommitted _camera** abgebremst wird, sondern gleich weich zur nächsten Position weiterfährt.

Hier wird die Zeitverzögerung mit der _camera camCommitPrepared 15 kombiniert um so eine weiche Fahrt zum Effekt zu haben. Am besten ihr probiert selbst einwenig mit diesen daten, um das für euch beste Ergebnis zu erzielen.
Hier ein kleines Beispiel, wie ein Teil unseres Scriptes aussehen könnte.

```
_camera camPrepareTarget Stryker
_camera camPreparePos [2839.51,2917.06,1.10]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 7
~6
_camera camPrepareTarget [16134.26,-96230.73,15.64]
_camera camPreparePos [2879.56,2886.94,0.99]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 15
~14
_camera camPrepareTarget Soldat
_camera camPreparePos [2884.51,2886.41,5.07]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 10
@camCommitted _camera
~4
TitleText ["Mein erstes Camscript ist fertig!","Plain Down"]
```

Kameraposition vorberechnen lassen

An diesem Beispiel möchte ich näher auf die Problematik des Texturnachladens bei schnellen Szenenwechseln eingehen. Viele werden es schon bemerkt haben, dass bei grösseren Positionssprüngen der Kamera in einer Sequenz, beim Start der Szene die Objekte und Texturen geladen werden, was einen doch sehr negativen Effekt zum Ausdruck bringt.
Um dem nun entgegenzuwirken, lassen wir einfach solche Szenen vorberechnen. Das geht ganz einfach mit PreloadCamera. Gebt einfach vor der Szene folgenden Befehl in euer Script ein:

```
_preload = [] spawn {waitUntil{preloadCamera [X,Y,Z]}}
@scriptDone _preload
```

Bei [XYZ] müst ihr noch die entsprechenden Koordinaten der _camera camPreparePos der Position eingeben. Jetzt wird die Szene vorberechnet und das Script wartet bis die Szene berechnet wurde. Somit ist dieser negative Effekt nicht mehr vorhanden.
Hier ein kleines Beispiel wie man diesen Befehl ins Scrript einfügen kann.

```
_camera camPrepareTarget Stryker
_camera camPreparePos [2839.51,2917.06,1.10]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 7
~6
_preload = [] spawn {waitUntil{preloadCamera [16134.26,-96230.73,15.64]}}
@scriptDone _preload
@camCommitted _camera

_camera camPrepareTarget [16134.26,-96230.73,15.64]
_camera camPreparePos [2879.56,2886.94,0.99]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
```

Kapitel 6: Kamera verfolgt Heli/Fahrzeug

Jetzt kommen wir zum etwas schwierigeren Teil. In unserer nächsten Szene werden wir die Kamera einen UH60 Blackhawk verfolgen lassen. Um das zu realisieren, spielen wir einwenig mit den X-Y-Z Koordinaten und erstellen so die Kamera mit Blick auf den fliegenden UH60, ohne dafür in den Kameramodus zu wechseln.Platziert nun einen Heli, nennt ihn **„UH60"** last ihn fliegen und gebt ihm einige Wegmarken vor, so dass er z.B. eine Patroullie fliegt. Speichert nun wieder eure Mission (Einleitung) ab.

Jetzt widmen wir uns dem Datenblock welchen wir nun selber schreiben. Seht ihn euch genauer an und ihr werdet merken, dass wir als Ziel der Kamera den UH60 eingegeben haben. Die Kamera wird also den Heli anvisieren, egal welche Position sie hat.

```
_camera camsettarget UH60
_camera camsetrelpos [-10,0,10]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
```

Abbildung 14: Kamera visiert Heli an.

Die Position bestimmen wir nun selber anhand unserer X-Y-Z Koordinaten, welche wir nun frei wählen um das bestmögliche Ergebnis zu erzielen. Die waagrechte Achse (X) lautet **-10**, die senkrechte Achse lautet **0**, und die Höhe der Kamera (X-Wert) steht auf **10**. Experimentiert selbst einwenig mit den Koordinaten für unterschiedliche Ergebnisse. Die X-Y Achsen lassen sich wunderbar kombinnieren.

Das hat zum Ergebnis: Die Kamera visiert den Heli von leicht rechts, Blick mitte Heli in einer Höhe von 10 Meter über dem Heli. Dürfte euch eigentlich soweit klar sein. ;-)
Jetzt visiert die Kamera zwar den Heli aus unserer gewählten Position an, folgt ihm aber noch nicht. Deshalb müssen wir nun den sogenannten **#Loop** (Label) einbauen. Der Loop steht für wiederholen eines Befehls, bis eine Bedingung den Loop wieder enden lässt. Für unser Beispiel wählte ich als Bedingung, das wenn der Fahrer kleiner/gleich als 5m an sein Auto kommt, der Loop beendet wird. Sehen wir uns nun unseren erweiterten Datenblock an. Der Fahrer hat nichts mit dem Heli zu tun. Er kommt in der nächsten Szene ins Spiel

```
#loop
_camera camsettarget UH60
_camera camsetrelpos [-10,0,10]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
? Fahrer distance Auto <=5 : goto "Endszene"
goto "loop"

#Endszene
titlecut [" ","BLACK Out",5]
~1
Player cameraEffect ["terminate","back"]
camDestroy _camera
ende1=true;
Exit;
```

Abbildung 15: Loop

Am Anfang unseres Kameradatenblocks steht das Label #Loop. Das heist jetzt startet ein Loop mit der kommenden Einblendung. Sobald nun die Kamera unseren Heli anvisiert, kommt der Befehl der Bedingung, den Loop auch wieder enden zu lassen (**? Fahrer distance Auto <=5 : goto "Endszene"**)
Heisst: Wenn (**?**) Fahrer **Distanz** zum Auto kleiner oder gleich 5 ist, dann (**:**) gehe zu (**goto**) „Endszene".
Heisst: Ist die Bedingung erfüllt, so liest das Script ab dem Label #Endszene weiter. Demnach endet dann unsere Szene. Ihr könnt natürlich irgendeine andere Bedingung eingeben. Die hier dient nur als Beispiel.
Danach kommt der Befehl goto „Loop". Das Script liest also die Szene, dann die Bedingung und dann den

Befehl goto „Loop". Das Script liest nun erneut die Szene, dann die Bedingung, dann wieder den Befehl goto „Loop". Die Kamera wird also solange immer den Loop ausführen und dem Heli folgen, bis die Bedingung erfüllt ist. Erst dann springt sie zum nächsten Label **#Endszene**. Die Szene wird dann beendet. Nun wollen wir unsere Verfolgungsszenen natürlich in unsere vorhandene scene.sqs eintragen und uns das Szenario mal ansehen.

```
titlecut [" ","BLACK IN",5]
_camera = "camera" camcreate [0,0,0]
_camera cameraeffect ["internal", "back"]

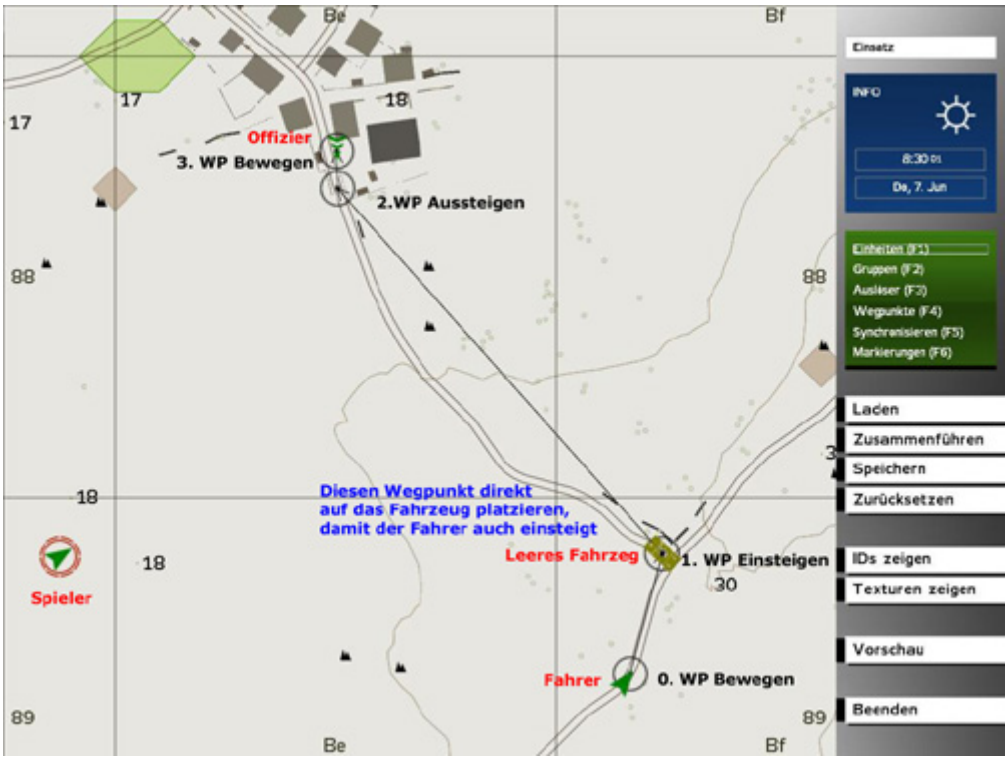

;=== 18:33:44--obere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2824.00,2917.12,10.27]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
~3
;=== 18:32:49--untere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2839.51,2917.06,1.10]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 7
@camCommitted _camera
~3
;=== 20:06:27--Blick Soldat im Dorf
_camera camPrepareTarget [16134.26,-96230.73,15.64]
_camera camPreparePos [2879.56,2886.94,0.99]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
~3
;=== 20:13:42---Blick Streife von oben im Dorf
_camera camPrepareTarget Soldat
_camera camPreparePos [2884.51,2886.41,5.07]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 6
@camCommitted _camera
~10

#loop
_camera camsettarget UH60
_camera camsetrelpos [-10,0,10]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
? Fahrer distance Auto <=5 : goto "Endszene"
goto "loop"

#Endszene
titlecut [" ","BLACK Out",5]
~1
Player cameraEffect ["terminate","back"]
camDestroy _camera
ende1=true;
Exit;
```

Abbildung 16: Fertiges Script mit Kamera die einem UH60 folgt

Damit das mit der Bedingung auch klappt, müsst ihr natürlich noch einen Fahrer platzieren. Ob das nun ein Zivilist ist, oder ein Soldat eurer Seite ist für dieses Beispiel völlig egal. Nennt ihn „**Fahrer**". Dann platziert ihr ein leeres Fahrzeug und nennt es „**Auto**"und weiter weg einen Zivilist oder Soldaten eurer Seite und nennt ihn „**Offizier**". Gebt nun dem Fahrer einen Wegpunkt direkt vor ihm, dann einen weiteren direkt auf das leere Fahrzeug. Hier wählt ihr den Wegpunkt-Typ „Einsteigen". Und nun noch einen weiteren Wegpunkt, in die Nähe des Offiziers, hier stellt ihr den Wegpunkt-Typ auf „Aussteigen. Den letzten Wegpunkt platziert ihr direkt vor dem Offizier. Es sollte alles nun in etwa so wie auf dem Beispielbild aussehen.



Ich mache dieses Beispiel deshalb so aufwendig, damit ihr auch gleich lernt, wie man jetzt z.B den Weg des Fahrers zum Auto zeitlich mit unserer Sequenz abstimmt. Denn je länger der Fahrer zum Auto braucht, desto länger ist die Kamera auf den Heli visiert. Unsere Sequent endet ja erst, wenn der Fahrer mindestens (weniger/gleich) 5m sich dem Auto nähert. Weshalb der Fahrer nun in das Auto einsteigen soll und zum Offizier fahren soll, hängt mit unserem nächsten Beispiel zusammen. Hier wird dann unsere Kamera das Auto verfolgen. Doch stimmt nun erstmal den Weg des Fahrers zum Auto mit unserer Sequenz ab ;-)

Ihr könnt auch die erste Wegmarke des Fahrers mit einem Auslöser synchronisieren und eine Bedingung eingeben (z.B: gofahrer), die ihr in unserem Script auf „true" stellen könnt. So geht der Fahrer erst los, sobald unser Script den Auslöser aktiviert. (gofahrer=true) So tut ihr euch mit der Abstimmung zwischen der sequenz und der Wegdauer des fahrers zum Auto leichter und könnt den fahrer starten lassen wann ihr wollt. ;-)

Hinweiss:
Für diese Beispiele habe ich eine Bespielmission beigefügt, damit ihr euch das ganze noch etwas genauer anschauen könnt.

Szenenwechsel bei Verfolgung

Nachdem ihr jetzt hoffentlich den Weg des Fahrers zum Auto mit unserer Sequenz abgestimmt habt, lassen wir nun die Kamera nachdem sie eine Zeit lang den Heli verfolgt hat, zur nächsten Szene weiterspringen. Jetzt kommt unser lieber Fahrer ins Spiel. Die Kamera wird nun an eine Position gesetzt, wo sie das Auto verfolgen wird, in das unser Fahrer einsteigen wird. Das ist ganz leicht und dürfte nun auch kein Problem mehr für euch sein.

Hierfür brauchen wir nun einen zweiten Loop, da ja die Kamera nun das Auto verfolgen soll. Wir schreiben nun eine weiteren Kameradatenblock, der nun wie folgt aussieht:

```
#Loop2
_camera camsettarget Auto
_camera camsetrelpos [8,-15,3]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
? Fahrer distance Offizier <10 : goto "Endszene"
goto "Loop2"
```

Abbildung 17: Loop2 wodurch das Auto von der Kamera verfolgt wird.

Wie ihr seht, ist das Ziel der Kamera nun das Auto. Die Position der Kamera hab ich hier frei Schnauze gewählt, findet hier selbst die für euch beste Einstellung.
Desweiteren ist nun auch eine neue Bedingung, welche zu „Endszene" wechselt, sobald der Fahrer sich weniger als 10 Meter den Offizier nähert.
? Fahrer distance Offizier <10 : goto "Endszene"
Was das Label #Endszene bewirkt dürfte euch noch klar sein. Diesen Koordinatenblock fügen wir nun direkt unter den Loop-Datenblock des Helis. Sehen wir uns nun das Ende des Scriptes an.

```
#Loop
_camera camsettarget UH60
_camera camsetrelpos [0,0,0]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
? Fahrer distance Auto <=5 : goto "Loop2"
goto "Loop"

#Loop2
_camera camsettarget Auto
_camera camsetrelpos [8,-15,3]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
? Fahrer distance Offizier <10 : goto "Endszene"
goto "Loop2"

#Endszene
titlecut [" ","BLACK Out",5]
~1
Player cameraEffect ["terminate","back"]
camDestroy _camera
ende1=true;
Exit;
```

Im Loop des Helis habe ich nun den goto-Befehl der Bedingung geändert, statt dass das Script zu #Endszene wechselt und Kamera beendet wird, wechselt sie nun zu Loop2. Verfolgt nun das Auto. Sehen wir uns in aller Ruhe nun das fertige Script ganz genau an. Dürfte jetzt eigentlich für jeden klar sein. ;-)

```
titlecut [" ","BLACK IN",5]
_camera = "camera" camcreate [0,0,0]
_camera cameraeffect ["internal", "back"]



;=== 18:33:44--obere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2824.00,2917.12,10.27]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
~3
;=== 18:32:49--untere Position
_camera camPrepareTarget Stryker
_camera camPreparePos [2839.51,2917.06,1.10]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 7
@camCommitted _camera
~3
;=== 20:06:27--Blick Soldat im Dorf
_camera camPrepareTarget [16134.26,-96230.73,15.64]
_camera camPreparePos [2879.56,2886.94,0.99]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
~3
;=== 20:13:42---Blick Streife von oben im Dorf
_camera camPrepareTarget Soldat
_camera camPreparePos [2884.51,2886.41,5.07]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 6
@camCommitted _camera
~6


#Loop
_camera camsettarget UH60
_camera camsetrelpos [0,0,0]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
? Fahrer distance Auto <=5 : goto "Loop2"
goto "Loop"


#Loop2
_camera camsettarget Auto
_camera camsetrelpos [8,-15,3]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
? Fahrer distance Offizier <=10 : goto "Endszene"
goto "Loop2"


#Endszene
titlecut [" ","BLACK Out",5]
~1
Player cameraEffect ["terminate","back"]
camDestroy _camera
ende1=true;
Exit;
```

Und schon wieder ist unser Script gewachsen, sehen wir uns das ganze nun im Editor an.

Kapitel 8: Test und Musik Einblendung

In unserem letzten Kapitel dieses Tutorials widmen wir uns der Text-und Musikeinblendung, um unsere Sequenz einwenig aufzuwerten. Was ist eine Szene ohne Musik und Text? Stimmt, nur was halbes. Befassen wir uns zuerst einmal mit den Texteinblendungen.

Texteinblendungen

Man hat verschiedene Arten von Möglichkeit um Texteinblendungen zu definieren. Dabei verwendet man folgende beiden Syntaxformen:

titleCut ["Hallo", "Black Out",5]

oder

TitleText ["Test", "Black In",5]

Diese beiden Beispiele nutzt man zur Einblendung bevor und nachdem der Bildschirm ein-und ausgeblendet wird.
"Black Out",5 bedeutet, dass der Bildschirm in einer Dauer von 5 Sekunden schwarz ausgeblendet wird.
"Black In",5 bedeutet genau das Gegenteil, hier wird der Bildschirm in einer Dauer von 5 Sekunden schwarz eingeblendet.

Es gibt mehrere Varianten, um Texteinblendungen darzustellen. Hier die Übersicht der möglichen Varianten:

Plain - Text erscheint mitten auf dem Bild
Plain Down - Text erscheint am unteren Bildrand
Black - blendet vom Bild ins Schwarze
Black Faded - blendet vom Bild ins Schwarze
Black In - blendet von schwarz ins Bild
Black Out - blendet Bild ins Schwarze aus
White In - blendet von weiß ins Bild
White Out - blendet Bild ins Weiße aus

Um jetzt beispielsweise einen Text mitten in der laufenden Sequenz einzublenden, bieten sich uns folgende Möglichkeiten. Die Texte werden so für ca. 10 Sekunden eingeblendet.

Hier ein kleines Beispiel dazu:

TitleText ["Mission was made by Imutep...","Plain Down"]

Hier ein Beispiel aus meiner Mission "Belle Foille" wie man einen Text ins Script einbindet:

```
;=== 19:48:13--Blick Palmen auf Stadt
_camera camPrepareTarget [-90162.82,26087.67,15.18]
_camera camPreparePos [7843.30,6221.43,6.45]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
~2
TitleText ["Mission was made by Imutep. . .","Plain Down"]
~5
;=== 19:52:28--Blick toter Zivi
_camera camPrepareTarget [-55015.42,84079.02,14.94]
_camera camPreparePos [7700.79,6190.12,0.69]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
```

Und so sieht das dann später ingame aus:



Die Texte werden ja für ca. 10 Sekunden eingeblendet. Wenn ihr jedoch einen Text kürzer als 10 Sekunden einblenden lassen wollt, weil es zeitlich sich so besser in die Szene integrieren lässt, so setzt einfach folgenden Befehl an die Stelle im Script, wo die Texteinblendung enden soll.

TitleText ["","Plain Down"]

Ein ganz einfacher Trick, weil hier das Script den Befehl bekommt einen anderen Text einzublenden und somit ja der vorige Text enden würde. Da kein Text eingegeben wurde, wird demnach auch nichts angezeigt. ;-)

Zeilenumbruch

Um einen Zeilenumbruch mit einzusetzen setzt man lediglich ein **\n** an die jeweilige Position im Text. Setzt man zwei **\n\n** hintereinander hat man eine Leerzeile und so weiter.

TitleText ["**Mission was made by Imutep...\n\nThanks for playing! ","Plain Down"]**



Musik einblenden

Jetzt wollen wir noch zum Abschluss ein Musikstück abspielen lassen, während unsere schwer erarbeitete Szene läuft. Auch hierfür brauchen wir nur einen kleinen Scriptbefehl.

playmusic "**Musikname**"
oder
playmusic „**Atrack24**";

Sehen wir uns den Befehl etwas genauer an:

playmusic ist der Befehl ein Musikstück abzuspielen.

"**Musikname**" Hier kommt der Name des Musikstücks rein, welches ihr in der entsprechenden Decsription.ext deklariert habt.

Atrack24 steht für ein Musikstück aus ArmA selbst. Hier die Nummer 24.

Wollt ihr einen Sound abspielen, so lautet der Befehl:
playsound "Soundname";

So würde nun der Anfang einer Sequenz aussehen. Die Musik soll hier schon zu Beginn der Mission abgespielt werden.

```
playmusic "zkouska";
_camera = "camera" camcreate [0,0,0]
_camera cameraeffect ["internal", "back"]

;=== 19:48:13--Blick Palmen auf Stadt
_camera camPrepareTarget [-90162.82,26087.67,15.18]
_camera camPreparePos [7843.30,6221.43,6.45]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
~2
TitleText ["Mission was made by Imutep. . .\n\nThanks for playing!","Plain Down"]
~5
TitleText ["","Plain Down"]
;=== 19:52:28--Blick toter Zivi
_camera camPrepareTarget [-55015.42,84079.02,14.94]
_camera camPreparePos [7700.79,6190.12,0.69]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
```

Musik ausblenden

Um jetzt zum Beispiel die Musik langsam ausblenden zu lassen, so können wir das ganz einfach mit diesem Befehl realisieren.

12 fademusic 0;

12 steht für die Zeit. In unserem Beispiel wird die Musik 12 Sekunden lang langsam ausgeblendet bis es schlieslich nach 12 Sekunden aus ist. Aus bedeutet hier ganz leise.

Fademusic bedeutet, die Musik lauter oder leider werden zu lassen. (Fade)

0 steht hier für die Lautstärke der abzuspielenden Musik. Wobei der Wert **0** für ganz leise steht und **1** ganz laut bedeutet. Schreibt einfach diesen Befehl an die Stelle im Script, an der die Musik ausgeblendet werden soll. Die Zeit wie lange die Ausblendung dauern soll, müsst ihr natürlich zeitlich mit eurem Script abstimmen, um ein optimales Ergebnis zu erzielen.

Hier nun unser fertiges Script, mit Musik und Texteinblendung:

```
playmusic "Musi1";
titlecut [" ","BLACK IN",5]
_camera = "camera" camcreate [0,0,0]
_camera cameraeffect ["internal", "back"]

_camera camPrepareTarget Stryker
_camera camPreparePos [2824.00,2917.12,10.27]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
~3
_camera camPrepareTarget Stryker
_camera camPreparePos [2839.51,2917.06,1.10]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 7
@camCommitted _camera
~3
_camera camPrepareTarget [16134.26,-96230.73,15.64]
_camera camPreparePos [2879.56,2886.94,0.99]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 0
@camCommitted _camera
~3
_camera camPrepareTarget Soldat
_camera camPreparePos [2884.51,2886.41,5.07]
_camera camPrepareFOV 0.700
_camera camCommitPrepared 6
@camCommitted _camera
~6
TitleText ["Mein erstes Camscript ist fertig!","Plain Down"]
~5
TitleText ["","Plain Down"]

#Loop
_camera camsettarget UH60
_camera camsetrelpos [10,-15,4]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
? Fahrer distance Auto <=5 : goto "Loop2"
goto "Loop"


#Loop2
_camera camsettarget Auto
_camera camsetrelpos [-8,-15,3]
_camera camPrepareFOV 0.700
_camera camCommit 0
@camCommitted _camera
? Fahrer distance Offizier <=10 : goto "Endszene"
goto "Loop2"


#Endszene
12 fademusic 0;
titlecut [" ","BLACK Out",12]
~1
Player cameraEffect ["terminate","back"]
camDestroy _camera
playmusic "";
0 fademusic 1;
ende1=true;
Exit;
```

Am Ende des Scriptes wieder ein Trick, um die Musik später wieder in originallautstärke zu haben.

```
Playmusic "";
0 fademusic 1;
```

Sonst würde man im Verlauf der Mission keine Musik mehr hören.
Zweite Möglichkeit wäre, ihr schreibt zu Beginn jeden Scriptes einfach 0 fademusic 1; rein.
Das hat demnach denselben Effekt.

Umgebungssound abstellen

Manchmal kann es in einer Sequenz störend wirken, wenn der Umgebungssound zu hören ist. Hierfür einfach an der bestimmten Position des Scriptes folgenden Befehl eingeben:

```
EnableEnvironment false;
```

Um den Umgebungssound wieder zu aktivieren, einfach folgenden Befehl eingeben:

```
EnableEnvironment true;
```

Funktexte deaktivieren

Um störende Funktexte in einer Sequenz abzustellen, einfach an der gewünschten Stelle folgenden Befehl eingeben:

```
EnableRadio false;
```

Zum aktivieren wieder auf true stellen.

```
EnableRadio true;
```

Kamera im Nachtsichtgerät-Modus

Um in der Sequenz die Kamera im Nachtsichgerät-Modus zu stellen, einfach folgenden Befehl im Script angeben:

```
CamUseNVG true;
```

Zum deaktivieren des Modus wieder folgenden Befehl eingeben:

```
CamUseNVG false;
```

Kamera im Fernglass-Modus

Es ist auch möglich, die Kamera im Fernglass-Modus zu stellen. Hierfür einfach folgenden Befehl im Script eingeben:

```
cutRsc ["binocular","PLAIN",3]
```

Hier wird die Kamera in einer Zeit von 3 Sekunden in den Fernglass-Modus gesetzt. Ein schöner Effekt, der jede Sequenz aufwerten kann wenn er gezielt eingesetzt wird.

DANKSAGUNG

Zum Abschluss möchte ich mich noch bei allen treu gebliebenen und neu dazugekommenen OFP-und ArmA Fans bedanken, ohne die es dieses Tutorial wohl nie gegeben hätte. Desweiteren gilt natürlich grosser Dank an Bohemia Interactive, die uns mit OFP und ArmA zwei erstklassige und umfangreiche Spiele lieferten, die uns auch noch sehr lange in den bann ziehen werden.

Und zuletzt möchte ich mich noch bei dem gesamten Team des **Assault Mission Studio** für die herzliche und gute Zusammenarbeit bedanken:

Stoned Boy, Onkel Bo, Cappa, M-E, und unserem ehemaligen Mitglied Marco-Polo-IV



CREDITS

Khartak

Mr.Murray

M-E

Stoned Boy

Sudden Death

http://www.bis.4players.de

http://www.mapfact.net

http://www.armedassault.com

http://www.flashpoint1985.com



AMS

http://www.assaultmissionstudio.de