

Config-Tools und CTW

Config-Tools bestehen aus:

- ◆ CfgCheck
- ◆ Bin2Cpp
 - ◆ Recon
 - ◆ Packer
- ◆ PBOInfo

CfgCheck

- Überprüft OFP-eigene Config-Dateien auf Fehler
- Config-Dateien sind: config.cpp, resource.cpp, description.ext, mission.sqm
- Gibt im Gegensatz zu OFP Art und Ort (Zeilennummer) des Fehlers an
- Binarisiert Config-Dateien (cpp → bin)
- CfgCheck besitzt einen an OFP angepassten C-Präprozessor, der auch #include in PBOs verarbeiten kann
- Kommandozeilen-Tool

CfgCheck

Wie wird CfgCheck verwendet?

- Drag&Drop: Einfach die gewünschte Datei auf die CfgCheck.exe ziehen und loslassen
- Kommandozeile:
`cfgcheck myconfig.cpp`

CfgCheck

CfgCheck liest automatisch den Pfad zur OFP-Installation aus. Damit können #include-Befehle aus PBO-Archiven korrekt interpretiert werden.

Da jeder seriöse Spieler/Modder Mod-Directories verwendet, unterstützt CfgCheck diese mit derselben Syntax wie die Resistance.exe:

```
cfgcheck -mod=map;cbt;rhs;laser;us  
description.ext
```

CfgCheck

Nochmal das Beispiel:

```
cfgcheck -mod=map;cbt;rhs;laser;us  
description.ext
```

Damit kann man testen, ob man in einer `description.ext` die `#include`-Befehle korrekt angewandt hat. Dies ist nützlich für Addons wie die F18 von Hudson/Pennywise, die für die Cluster-Bombs einen Dialog verwendet. Fehlerhafte `include`-Befehle ergeben einen Crash!

CfgCheck

```
class SCFW_MARS : M113
00110101 01110101
VehicleClass = "SCF Bundeswehr Vehicles";
10111011 11111110
model = "scfw_mars\scfw_mars_1.psd";
displayname = "MARS";
01011101 10001111
mapSize = 8.0; // 8 * 4 meter in Origen
10001110 10101111
color = "rgb(100,100,100)"; // map icon
picture = "scfw_marsvumars_01.psd"; // command bar
weapons[] = { "SCF_MARSLauncher" };
00110101 01110101
magazines[] = { "SCF_MARSLauncher" };
10111011 11111110
casFaction = false;
hasCommander = false;
01011101 10001111
hasGunner = true;
hasDriver = true;
10001110 10101111
secondaryExplosion = 1;
crew = "BMOD_CrewT";
```

- Binarisierung von Configs

cfgcheck -bin myconfig.cpp

- „Verschönerte“ (beautified) Configs

cfgcheck -b myconfig.cpp newconfig.cpp

- Um Tabulatoren für die Einrückung zu verwenden: -tabs
- Kommentare nach jeder Klasse: -cc

Bin2CPP

Konvertiert BIN-Dateien (binarisierte Configs) zu text-basierten Configs.

Bin2CPP

- Debinarisiert eine Config-Datei, so daß sie für Menschen lesbar ist
- Extrahiert aus PBOs automatisch die config.cpp, oder konvertiert eine enthaltene config.bin zu einer config.cpp
- Z.B.: M2A2.pbo → M2A2.cpp
- Kommandozeilen-Tool

```
bin2cpp config.bin -o config.cpp
```

Hat genauso wie CfgCheck: -tabs, -cc

Recon

Das voll-automatisierte Addon-Erstellungs-
Werkzeug RECON

Recon

- Automatisiert einige Abläufe bei der Arbeit an einem Addon:
 - Config-Überprüfung (CfgCheck-Engine)
 - Aufruf von Binarize (P3D-Überprüfung)
 - Packen in eine PBO
 - Kopieren in gewünschtes Addon-Verzeichnis
- Erzeugung von Informationen über die PBO:
 - Versionsnummer wird in version.sqf abgespeichert
 - Versionsnummer und Datum in build.numDiese Informationen können mit PBOInfo ausgelesen werden. Evtl. kann damit eine Addon-Verwaltung aufgebaut werden

Recon

- Beispiel-Aufruf

```
recon rad_f18 us -hwtl -binconf
```

- rad_f18: Verzeichnis des zu erzeugenden Addons
- us: Mod-Verzeichnis (rad_f18.pbo wird kopiert nach OFP/us/addons)
- -hwtl: Modelle werden für Hardware Transform&Lighting optimiert
- -binconf: Config.cpp wird binarisiert

Recon

Binarize ist ein sehr nützliches Tool, um Addons zu optimieren. Es kopiert nur genau die Texturen, die auch von einem Modell verwendet werden.

Deshalb sollte man für Texturen, die von keinem Modell direkt verwendet werden (z.B. Blood-Textures) oder Interface-Symbole, Dummy-Modelle erstellen, die diese in das Opt-Verzeichnis kopieren

Dies ist besser, als „indirekte“ Texturen per Hand zu kopieren, da man das oft vergißt.

Packer

- Packer ist ein einfaches PBO-Erzeugungstool
- Es erhält ein Verzeichnis, erzeugt eine PBO und kopiert es in ein bestimmtes Verzeichnis eines bestimmten Mods
- Beispiel:

```
packer sounds mymod dta
```

Erzeugt eine sounds.pbo und kopiert sie nach
OFP/mymod/dta/sounds.pbo

PBOInfo

- Liest die von Recon/Packer erzeugte build.num aus und zeigt die Informationen an (Versionsnummer und Erzeugungsdatum)
- Berechnet bei Bedarf die md5-Summe

Conquer The World „CTW“

- CTW ist eine Sammlung von Content-Generatoren für OFP/ArmA
 - Missions-Generator
 - Insel-Generator
 - Evtl. Kampagnen-Generator
- ArmA-Szenarien (Mission+Insel) sollen abhängig vom CTW-Interface erzeugt werden
- Dieses Interface sollte eine Art Weltkarte sein

Mission Generator

- Erzeugt Missionen automatisch aus Config-artigen Dateien
- Analysiert Inseln und findet strategisch wichtige Ziele
- Spezielle Objekte und Inseltexturen werden zu sogenannten Clustern zusammengefasst und als mögliche Ziele definiert
- Z.B. bestehen Stadt-Cluster aus Häusern, Schulen, Kirchen; Farmen aus Farmgebäuden und Feld-Texturen

Mission Generator

- Um realistische Szenarien zu erstellen, wird dem Generator ein ORBAT übergeben (Order of Battle)
- Diese gibt die Struktur der beiden Streitkräfte an
- Dabei gibt es BLUFOR (die Seite des Spielers) und die OPFOR (Gegnerseite)
- Aus dem ORBAT werden automatisch Einheiten erstellt (Regimenter, Kompanien, Züge, Gruppen), je nachdem, was in der Config spezifiziert wurde.
- Diesen werden Tasks zugeordnet: Angriff, Nachschub, Unterstützung, Aufklärung, Abwehr

Mission Generator

- Eine automatische Einbindung von GroupLink und/oder DAC ist denkbar
- Einbindung von Addons ist extrem einfach: In die ORBAT-Config muß der CfgPatches-Name eingetragen werden; und es kann in eine beliebige Einheit der Klassenname (wie „SoldierWB“) eingetragen werden

Mission Generator

- Beiden Seiten kann Stärke (0-100%) und Skill (0-100%) zugewiesen werden. Stärke bedeutet wieviel der Einheiten verfügbar sind; Skill stellt den OFP-eigenen Skill der Einheiten ein
- Es kann festgelegt werden, welcher Teil und wieviel der Insel von welcher Seite dominiert wird. z.B. BLUFOR im Westen, OPFOR im Osten
- Es kann Typ der Mission festgelegt werden: Rückzug, Verteidigung, Meeting, Angriff und Jagd

Mission Generator

- Zusätzlich zu den Gruppen werden noch Basen erzeugt
- Basen können einfach in OFP im Editor erstellt werden; die Objekte können über ein Skript ausgelesen werden und geben eine Klassen-Definition aus, die in die Missions-Config eingefügt werden kann
- Diese dienen als Ausgangs- oder Versorgungspunkt

Mission Generator

- Ein einigermaßen realistisches Briefing wird automatisch erstellt
- Jede Einheit der eigenen Seite erhält dabei eine ID (z.B. OB92) und einen Marker
- Auf der Karte werden die vermuteten gegnerischen Stellungen eingezeichnet, sowie deren Ausrüstung im Briefing angedeutet

Island Generator „ATLAS“

Automated Terrain, Landscape and Artefact System

- Erzeugt ein Höhenfeld (Heightmap)
- Verteilt Texturen aufgrund Höhe/Gefälle und Zufall
- Verteilt Bäume und Büsche auf der ganzen Insel, sowie Wälder
- Unterstützt Cell-Grids mit einem Abstand von weniger als 50m

CTW Content Generators

