

**Sternenschiff Catan**  
**PC CD-ROM**

**Handbuch Scripting**  
**Deutsch**

Version 1.1  
28. November 2003

## **Impressum**

Kein Teil des Inhalts, ob teilweise oder vollständig, darf kopiert, realisiert, programmiert, produziert, gesendet, in eine andere Sprache übersetzt oder übertragen werden ohne ausdrückliche, schriftliche Genehmigung von drei partner.

drei partner behält sich vor, das in diesem Handbuch beschriebene Produkt zu jeder Zeit und ohne Vorankündigung zu ändern.

Software, Idee, Konzept, Text, Grafik und Namen sind durch internationales Copyright © geschützt.

Dieses Produkt ist durch die allgemein für Computersoftware geltenden Urheberrechtsgesetze geschützt. Sie dürfen die Software nicht kopieren.

© 2002-2003 drei partner, [www.dreipartner.de](http://www.dreipartner.de)

© 2002-2003 Catan GmbH, [www.catan.com](http://www.catan.com) [www.klausteuber.de](http://www.klausteuber.de)

© 2003 Take 2 Interactive GmbH, [www.take2.de](http://www.take2.de)

Alle Rechte vorbehalten. All rights reserved.

## **Installation**

Sie können mit dem Editieren beginnen, sobald Sie Sternenschiff Catan installiert haben. Das Spiel benötigt die "Sternenschiff Catan" CD im Laufwerk während des Spielens. Die Systemvoraussetzungen, um das Produkt spielen zu können, entnehmen Sie bitte der Packung.

Sobald Sie das automatische Update System ausgeführt haben, stehen alle Dateien zur Verfügung die benötigt werden, um Sternenschiff Catan Scripte zu entwerfen.

Zusätzlich wird ein externer Text Editor benötigt. Die Script Dateien sind im TXT Text Format abgelegt. Andere Formate (etwa ein Textverarbeitungsformat) behindern oder blockieren die Ausführung.

## **Modding**

Der Begriff "modding" wird bei vielen Spielen für die externe Veränderung des Spielinhaltes durch verschiedene Techniken, bis zur kompletten grafischen Umarbeitung benutzt. Ein "Mod" verändert das Programm und kann aus mehreren Dateien bestehen.

Die Software Sternenschiff Catan verfügt über die Möglichkeit, extern viele Spielwerte, praktisch das gesamte Regelwerk zu verändern und die Werte neu einzustellen.

Sie können neue Karten entwerfen, Szenarien aufbauen und ganz neue Missionen entwerfen. Dazu müssen sogenannte Scripts entworfen werden, die das Spiel in einem speziellen Modus verarbeiten kann.

Sie finden im Ordner "missions" diese Anleitung, sowie Beispieldateien.

Wir wünschen viel Spass beim basteln, müssen aber auch darauf hinweisen, dass für diese Option kein Kundendienst zur Verfügung steht. Bitte haben Sie hierfür Verständnis.

## **Warnung**

Die Scripte greifen tief in das Spielsystem von Sternenschiff Catan ein und können das gesamte Spiel, dessen Design und Mechanik völlig verändern.

Bitte beachten Sie, dass falsche Befehle, Parameter oder auch andere Bedingungen das Spiel zu einem Absturz bringen können. Sternenschiff Catan ist in der Lage viele Fehlerquellen zu erkennen, zu melden und zu umgehen, jedoch ist die richtige Editierung der Scripte in Form und Funktion für einen Erfolg ausschlaggebend.

Ebenso sind die Abstimmung der Karten und dessen Werte nach einem speziellen Muster und Design vorgenommen worden. Wir empfehlen für den Beginn die bestehenden Scripte zu verwenden und diese zu verändern, bevor komplett neue System aufgebaut werden.

## **Starten des Spiels**

Das Spiel startet automatisch sobald die CD eingelegt wird den Startbildschirm. Über den Menüpunkte "Sternenschiff Catan" wird das Auswahlmeneu „Startprogramm“ gestartet.

Sie können das Spiel jederzeit über das Startmenu oder über das Desktop Icon starten. Das Spiel kann auch direkt aus dem Ordner, in dem es installiert wurde, gestartet werden, ein Doppelclick auf "Sternenschiff Catan" reicht dazu aus.

Die CD muß zum Spielen eingelegt sein.

## **Scripting**

### Gross/Kleinschreibung

Alle Scripte beachten die Gross/Kleinschreibung auch in der Ausführung. Der Bezeichner "Hello" ist also unterschiedlich vom Bezeichner "HELLO" !

### Besondere Zeichen

Für das Scripting sollte auf besondere Zeichen verzichtet werden. Für alle Variablen, Befehle und Objekte sowie alle anderen Elemente sollte ausschließlich der Satz der Buchstaben 'a' bis 'z', 'A' bis 'Z' und der Zahlen '0' bis '9' verwendet werden.

### Kommentare

Kommentare können zu jeder Zeit gesetzt werden, sie sind repräsentiert durch zwei aufeinanderfolgende Minuszeichen '--'. Alles dahinter wird ignoriert.

### Reservierte Worte

Alle Befehle sowie das Wort 'self' sollten nicht als Variablen eingesetzt werden.

## **Interpretation**

### Single Pass System

Das System verfügt über einen Durchlauf, in dem alle Variablen deklariert werden. Es kann also nur auf bereits deklarierte Werte zugegriffen werden. Alle Variablen die später als zur ausführenden Zeile deklariert werden sind bis dahin unbekannt.

Dies ist wichtig bei Definition der Karten, es muss auf eine bestimmte Reihenfolge geachtet werden.

## **Script Ausführung**

Die Funktion "ExecuteRegisterScript" wird vom Spiel aus bei Beginn

automatisch ausgeführt. Hier liegt der Startup Code. Alle Variablen stehen zu diesem Zeitpunkt zur Verfügung. Interaktive Berechnungen und Initialisierungen zu Beginn des Spiels können hier eingefügt werden.

```
"  
function ExecuteRegisterScript( )  
  
-- content  
  
end  
"
```

## **Regeln zum Entwurf der Karten**

### **Kartendefinition**

Den Kern jedes Spiels bildet die Kartenliste. Jede Karte wird durch bestimmte Werte und Funktionen definiert. Dabei ist jedoch zu beachten, dass jede Karte nur einmal im Spiel verwendet werden darf. Um mehrere identische Karten zu erstellen, müssen Sie die Kartendefinition mit jeweils anderen Kartennamen mehrmals kopieren.

Das Script der Kartenliste wird zu Beginn einer Mission einmal ausgeführt. Zu dem Zeitpunkt werden die Kartenwerte in das Spiel übernommen. Einige der definierten Funktionen einer Karte können auch während eines Spieles aufgerufen werden.

#### Ein wichtiger Hinweis:

Ein gutes Kartenset für eine Mission zu erstellen ist ein relativ aufwendiger Prozess, der viele Stunden testen benötigt. Oftmals haben bestimmte Wertekombinationen mehrerer Karten einen Hintergrund in der Spielmechanik.

Wir empfehlen zu Beginn erstmal die mitgelieferten Beispielscripte als Basis zu verwenden und mit kleinen Änderungen deren Einfluss auf das Spiel zu testen.

## Kartendesign

Bei den Abenteuerkarten ist zu beachten, dass jeder Karte immer nur eine Funktion zugeordnet sein darf. Ebenso darf bei den Zielplaneten keine Überschneidung von Zielen, Zentrumsaktionen und Spezialaktionen vorkommen.

Es ist im Spiel nicht möglich mehr als 3 Aktionen pro Zielplanet gleichzeitig aktiv zu haben, daher kann eine Mehrfachbelegung mit Funktionen zu einem undefinierten Verhalten des Spiels führen.

## Kartenwerte

- name

Der Name der Karte wird mehrfach im Programm verwendet. Zum einen wird dadurch eine Referenz auf einen definierten Namen der Spieltexte gebildet, zum anderen ist dieser Name eine eindeutige Bezeichnung, über die im Programm und in anderen Scripten auf diese Karte zugegriffen wird. Der Name muss aus dem Pool vorhandener Kartennamen gewählt werden und darf nur einmal pro Kartenliste verwendet werden.

- defaultstack

Dies ist die Bezeichnung des Standardstapels, auf den die Karte bei der Initialialisierung durch den Befehl `SetAllCardsToDefaultStack()` gelegt werden. Die möglichen Namen entnehmen Sie dem Anhang.

- classname

Dieses Feld MUSS die genaue Bezeichnung der Karte im Script enthalten, da das Spiel darüber intern auf einige der Werte zugreift. Also z.B. für die Karte `card.pallas` muss definiert sein `classname = "card.pallas"`.

- siegpunkt

- freundschaft

- orden

Diese Werte legen fest, wieviele Siegpunkte, Freundschaftspunkte oder Orden ein Spieler durch diese Karte erhält, wenn er sie bekommt.

- video\_approach

- video\_loop

Diese beiden Felder enthalten die Ressourcennamen der Videos, die auf der Karte beim Anflug abgespielt werden sollen. Leere Einträge ("" ) können auch verwendet werden, um keine Anflüge auf der Karte zu zeigen.

– gfx\_still

Dies ist der Ressourcenname der Kartengrafik.

Beispieldefinition einer Karte:

```
"
-- define CARD_PALLAS (2)
card.pallas =
{
  name = "CN_PALLAS",
  defaultstack = "stack_tmpsektor",
  classname = "card.pallas",
  siegpunkt = 0,
  freundschaft = 0,
  orden = 0,

  video_approach = "",
  video_loop = "",
  gfx_still = "",

  DefineAction = function ( self )
    end,
  Approaching = function ( plrnumber )
    end,
}
"
```

## Kartenfunktionen

– DefineAction

Dies ist die wichtigste Funktion zur Definition einer Karte. Hier werden alle Eigenschaften und Funktionen einer Karte durch sogenannte "Attach"-Befehle definiert. Durch die Kombination von diesen Befehlen in der Funktion sind sehr komplexe Karten möglich. Beachten Sie aber, dass



bestimmte Kombinationen keinen Sinn machen und zu Problemen im Spiel führen können.

```
"  
-- define card behaviour  
DefineAction = function ( self )  
  
end,  
"
```

#### – Approaching

Diese Funktion einer Karte wird aufgerufen, wenn diese im Spiel angefliegen wird. Als Parameter wird die Spielernummer übergeben.

```
"  
-- define script when approaching  
Approaching = function ( plrnumber )  
  
end,  
"
```

### **Interne Kartenfunktionen**

Da die internen Kartenfunktionen immer im Kontext einer Kartendefinition ausgeführt werden, muss als erster Parameter immer der Parameter "self" angegeben werden. Dieser Parameter ist in der Funktion DefineAction() einer Karte automatisch richtig gesetzt.

## ANHANG 1: Tabellen

### Stapel (stacks)

Name	Type	Beschreibung, Karten Maximum
"stack_main"	Intern !	Nutzung nicht erlaubt 256
"stack_nirvana"	game	Zwischenspeicher Stapel 256
"stack_sektormain_0"	game	Sektor Stapel I 25
"stack_sektormain_1"	game	Sektor Stapel II 25
"stack_sektormain_2"	game	Sektor Stapel III 25
"stack_sektormain_3"	game	Sektor Stapel IV 25
"stack_sektorreserve"	game	Sektor Reserve Karten 25
"stack_player_1"	game	Spieler 1 Handkarten 30
"stack_player_2"	game	Spieler 2 Handkarten 30
"stack_player_1_center"	game	Spieler 1 Zentrumskarten 6
"stack_player_2_center"	game	Spieler 2 Zentrumskarten 6
"stack_adventure_1"	game	Abenteuerkarten, Stapel I 20
"stack_adventure_2"	game	Abenteuerkarten, Stapel II 20
"stack_adventure_3"	game	Abenteuerkarten, Stapel III 20
"stack_adventure_reserve"	game	Abenteuerkarten, Stapel Reserve 256
"stack_tmpsektor"	helper	Globaler Sektor Arbeitsstapel 256
"stack_tmpsektorreserve_1"	helper	Sektor Arbeitsstapel "I" 25
"stack_tmpsektorreserve_2"	helper	Sektor Arbeitsstapel "II" 25
"stack_tmpsektorreserve_3"	helper	Sektor Arbeitsstapel "III" 25
"stack_tmpsektorreserve_4"	helper	Sektor Arbeitsstapel "IV" 25
"stack_flight_actual"	game	Flugreihe aktueller Spieler 25
"stack_tmpadventure_1"	helper	Arbeitsstapel Abenteuerkarten 1 25
"stack_tmpadventure_2"	helper	Arbeitsstapel Abenteuerkarten 2 25
"stack_tmpadventure_3"	helper	Arbeitsstapel Abenteuerkarten 3 25
"stack_tmpadventure_4"	helper	Arbeitsstapel Abenteuerkarten 4 25
"stack_tmp_1"	helper	Temporärer Arbeitsstapel 1 25
"stack_tmp_2"	helper	Temporärer Arbeitsstapel 2 25
"stack_tmp_3"	helper	Temporärer Arbeitsstapel 3 25
"stack_tmp_4"	helper	Temporärer Arbeitsstapel 4 25
"stack_tmp_5"	helper	Temporärer Arbeitsstapel 5 25
"stack_center"	game	Verfügbare Zentrumskarten 25
"stack_center_player_1"	game	Start Zentrumskarten, Spieler 1 10
"stack_center_player_2"	game	Start Zentrumskarten, Spieler 2 10
"stack_ablage"	game	Ablagestapel 256
"stack_cardaction"	game	Spezielle Aktions Karten 50

## **Erläuterung Stapel Tabelle**

Das Spiel arbeitet mit verschiedenen Stapeln, welche sehr sinnvoll für den Spielaufbau und auch analog zum Brettspielaufbau funktionieren. Sie sind aufgeteilt in Hilfsstapel (helper) und Spielstapel (game).

Die Stapel arbeiten nach dem "first in - first out" Prinzip, eine Karte die zuerst auf einen Stapel bewegt wird, wird auch vom Spiel zuerst von dort entnommen. Somit besteht die Garantie, für einen seriellen Aufbau von Spielszenarien.

Die Stapel werden von den Befehlen mit ihren in der ersten Spalte angegebenen Namen adressiert und sie sind deshalb in Anführungszeichen.

Die Hilfsstapel sind Hilfen für die Erstellung der Szenarien, ihre Namen sind nicht bindend, sie wurden als reine Erleichterung bei der Erstellung entworfen und können frei benutzt werden.

Jeder Stapel verfügt über eine Grenze, bis zu der Karten eingefügt werden können. Wird darüber hinaus gearbeitet werden diese Karten ignoriert und nicht eingefügt.

Der Stapel "Main" darf extern nicht benutzt werden, er enthält die komplette Spielkartenliste.

## STATES

"invalid"  
"gamestart"  
"roundstart"  
"productionroll"  
"productionselect"  
"flightsector"  
"flight"  
"fight"  
"buildtrade"  
"updatecenter"  
"variabletrade",  
"planettrade",  
"doplanettrade"  
"resourcejoker"  
"task"  
"selectcenter"  
"playertrade"  
"scancenter"  
"servicecenter"  
"playerinfo"  
"selectdice"  
"roundend"  
"gameend"

## SUBSTATES

"none"  
"prod\_dice\_running"  
"prod\_center\_production"  
"prod\_select\_start"  
"prod\_next\_player"  
"prod\_execute\_production"  
"flight\_start"  
"flight\_approaching"  
"flight\_next\_action"  
"flight\_select\_action"  
"flight\_continue"  
"flight\_reinit"  
"payordie\_init"  
"payordie\_select"  
"result\_init"  
"result\_wait\_forok"  
"result\_wait\_for\_show\_end"  
"result\_check\_and\_activate"  
"result\_finish"  
"result\_dice\_roll\_player"  
"result\_dice\_roll\_player\_running"  
"result\_dice\_roll\_pirate"  
"result\_dice\_roll\_pirate\_running"  
"init",  
"build\_menu"  
"select"  
"finish"  
"wait\_msg"  
"starmoon\_start"  
"starmoon\_running"  
"show\_symbol"  
"select\_center\_init\_next"  
"multiplayer\_wait"

## **Zentren (center type names)**

"logistics"  
"scan"  
"science"  
"admin"  
"production"  
"trade"  
"transport"  
"service"  
"analyse"

## **Rohstoffe (resource names)**

"erz"  
"treibstoff"  
"nahrung"  
"carbon"  
"handelsware"  
"wissen"

## **ANHANG 2: Befehlsliste, Script Befehle**

### **Übersicht**

Folgende Befehle stehen zur Verfügung:  
(Gelistet mit Syntax zur Kurzreferenz)

```
AddAim_Astro( PLAYER_ID, ASTRO, "result" )
AddAim_MostTransport( PLAYER_ID, WINPOINTS, "result" )
AddAim_MostTransportAtRound( PLAYER_ID, ROUND, "result" )
AddAim_Winpoints( PLAYER_ID, WINPOINTS, "result" )
AddAim_MostAstroAtRound( PLAYER_ID, ROUND, "result" )
AddAim_MostWinpointsAtRound( PLAYER_ID, ROUND, "result" )
AddAim_ReachRound( PLAYER_ID, ROUND, "result" )
DumpCardList()
DumpCardStack("STACKNAME")
InsertCardListToIndex( CARDLIST_OBJECT )
MoveCardToStack( "CARD NAME", "DESTINATION STACKNAME" )
MoveMultipleCardsToStack( "SOURCE STACKNAME", "DESTINATION STACKNAME", COUNT )
ReplaceCard( "REPLACED CARDNAME", "NEWCARD CARDNAME" )
SetAllCardsToDefaultStack()
ShuffleStack( "STACKNAME" )
SetStackStatus( "STACKNAME", status )
```

## Befehlsliste

### AddAim\_Astro

#### FUNCTION

Fügt ein Ziel für einen Spieler hinzu, das bei einem erreichten Astro Betrag das Ergebnis Gewonnen "won" oder Verloren "lost" setzt.

#### SYNOPSIS

AddAim\_Astro( PLAYER\_ID, ASTRO, "result" )

result: "won", "lost"

#### DEMO

AddAim\_Astro( 0, 35, "won" )

Dies ist ein Ziel für Spieler 0 (eins im Spiel), bei 35 erreichten Astro gewinnt er.

### AddAim\_MostTransport

#### FUNCTION

Fügt ein Ziel für einen Spieler hinzu, das bei einem erreichten Betrag an, mit dem Transport Zentrum, transportierten Waren das Ergebnis Gewonnen "won" oder Verloren "lost" setzt.

#### SYNOPSIS

AddAim\_MostTransport( PLAYER\_ID, WINPOINTS, "result" )

result: "won", "lost"

#### DEMO

AddAim\_MostTransport( 1, 13, "won" )

Dies ist ein Ziel für Spieler 0 (zwei im Spiel), bei 13 erreichten Transporteinheiten wird er das Spiel gewinnen.

### AddAim\_MostTransportAtRound



## FUNCTION

Dieser Befehl fügt ein Ziel hinzu, das die transportierten Waren in einer bestimmten Runde misst, sind dies mehr als bei seinem Mitspieler, wird das Ergebnis "result" gesetzt.

## SYNOPSIS

```
AddAim_MostTransportAtRound( PLAYER_ID, ROUND, "result" )
```

result : "won" , "lost"

## DEMO

```
AddAim_MostTransportAtRound( 1, 24, "won" )
```

Dies ist ein Ziel für Spieler 0 (zwei im Spiel), hat er in Runde 24 die meisten Transporteinheiten insgesamt wird er das Spiel gewinnen.

## AddAim\_Winpoints

## FUNCTION

Fügt ein Ziel vom Typ Siegpunkte (winpoints) hinzu. Dieses Ziel ist das gesamte Spiel aktiv und wird ausgelöst sobald die Anzahl erreicht ist.

## SYNOPSIS

```
AddAim_Winpoints( PLAYER_ID, WINPOINTS, "result" )
```

result : "won" , "lost"

## DEMO

```
AddAim_Winpoints( 1, 10, "won" )
```

Dies ist ein Ziel für Spieler 1 (zwei), bei 10 Siegpunkten gewinnt er das Spiel.

## AddAim\_MostAstroAtRound

## FUNCTION

Fügt ein Ziel für einen Spieler hinzu, das die meisten erreichten Astro ab einer Runde festhält.

## SYNOPSIS

```
AddAim_MostAstroAtRound( PLAYER_ID, ROUND, "result" )
```

result : "won" , "lost"

## DEMO

```
AddAim_MostAstroAtRound( 0, 15, "won" )
```

Dieses Ziel für Spieler 0 (eins) läßt ihn gewinnen wenn er ab Runde 15 die meisten Astro erhält.

## AddAim\_MostWinpointsAtRound

## FUNCTION

Fügt ein Ziel für einen Spieler hinzu, das die meisten erreichten Siegpunkte in einer Runde festhält.

## SYNOPSIS

```
AddAim_MostWinpointsAtRound( PLAYER_ID, ROUND, "result" )
```

result : "won", "lost"

## DEMO

```
AddAim_MostWinpointsAtRound( 1, 27, "lost" )
```

## AddAim\_ReachRound

## FUNCTION

Setzt ein Ziel, das bei Erreichen einer Runde ausgelöst wird. Dies wird in jedem Fall automatisch durchgeführt.

## SYNOPSIS

```
AddAim_ReachRound( PLAYER_ID, ROUND, "result" )
```

result : "won", "lost"

## DEMO

```
AddAim_ReachRound( 0, 12, "won" )
```

"

This is an aim for player 0, reaching round 12 he wins the game (Player ids starting at zero.)

## DumpCardList

## FUNCTION

Gibt alle registrierten Karten in das LOGFILE aus. Anhand der Werte kann überprüft werden ob Attach Befehle richtig ausgeführt wurden.

## SYNOPSIS

DumpCardList()

## DEMO

DumpCardList()

## DumpCardStack

## FUNCTION

Gibt alle Karten dieses Stapels in das LOGFILE aus.

## SYNOPSIS

DumpCardStack("STACKNAME")

## DEMO

DumpCardStack("stack\_tmpsektor")

## InsertCardListToIndex

## FUNCTION

Fügt eine Tabelle mit den Namen von Kartenobjekten zum Spielindex hinzu. Diese Karten werden dem Spiel als bekannt gemeldet und gleichzeitig initialisiert. Diese Funktion kann mehr als einmal aufgerufen werden um größere Sinngruppen an Kartendefinitionen zu registrieren.

## SYNOPSIS

InsertCardListToIndex( CARDLIST\_OBJECT )

## DEMO

```
cardlist =  
{  
  card.one,  
  card.two,  
}
```

InsertCardListToIndex( cardlist )

## **MoveCardToStack**

### **FUNCTION**

Bewegt eine einzelne Karte auf einen Stapel. Der Parameter in der Kartendefinition "name = " wird hierbei benutzt.

### **SYNOPSIS**

MoveCardToStack( "CARD NAME", "DESTINATION STACKNAME" )

### **DEMO**

MoveCardToStack( "CN\_HADES", "stack\_tmpsektor" )

## **MoveMutipleCardsToStack**

### **FUNCTION**

Bewegt eine Anzahl von Karten von einem Stapel zu einem anderen. Die ersten n Karten werden auf den Zielstapel bewegt. Hinweis: eine einzelne Karte kann mit COUNT=1 bewegt werden.

### **SYNOPSIS**

MoveMultipleCardsToStack( "SOURCE STACKNAME", "DESTINATION STACKNAME", COUNT )

### **DEMO**

MoveMultipleCardsToStack( "stack\_tmpsektor", "stack\_sektormain\_0", 10 )

## **ReplaceCard**

### **FUNCTION**

Ersetzt die angegebene Karte mit einer anderen. Die erste Karte wird auf den Stapel "stack\_nirvana" abgelegt. Der Parameter in der Kartendefinition "name = " wird hierbei benutzt.

### **SYNOPSIS**

ReplaceCard( "REPLACED CARDNAME", "NEWCARD CARDNAME" )

### **DEMO**

ReplaceCard( "CN\_HADES", "CN\_PALLAS" )

## **SetAllCardsToDefaultStack**

### **FUNCTION**

Kopiert alle Karten zu ihren Standard Start Positionen, die im Parameter "defaultstack = " der Kartendefinition angegeben sind. Diese Funktion sollte einmal nach der Initialisierung aufgerufen werden und baut viele Stapel vorbereitet auf.

### **SYNOPSIS**

SetAllCardsToDefaultStack()

### **DEMO**

SetAllCardsToDefaultStack()

## **ShuffleStack**

### **FUNCTION**

Mischt alle Karten in diesem Stapel.

### **SYNOPSIS**

ShuffleStack( "STACKNAME" )

STACKNAME

Der vorderfinierte Name des Stapels aus der Stapelliste.

### **DEMO**

ShuffleStack( "stack\_tmpsektor" )

## **SetStackStatus**

### **FUNCTION**

Setzt die Statuswerte für einen Stapel (stack) und legt fest ob und ab wann dieser Stapel neu gemischt wird.

### **SYNOPSIS**

SetStackStatus( "STACKNAME", status )

STACKNAME

Der vorderfinierte Name des Stapels aus der Stapelliste.

Wir das Schlüsselwort "all" angegeben, gilt diese Einstellung für ALLE vorhandenen Stapel.

status

Werte: -1 = Stapel wird nicht gemischt

0 = Stapel wird bei Rundenstart neu gemischt. (Vorgabe)

>0 = Ab dieser Runde wird neu gemischt (einschließlich)

## DEMO

SetStackStatus( "stack\_tmpsektor", -1 )

## ANHANG 3: Befehlsliste, Objekt Befehle

### Übersicht

AttachAddAction( SELFKEYWORD, ACTION\_NUM )  
AttachAddStore( SELFKEYWORD, STORE\_NUM )  
AttachBordStateComment( SELFKEYWORD, "state", "substate", CALLED, TURN, "text", SPEECH\_TAKE )  
AttachCardText( SELFKEYWORD, "resource name" )  
AttachCardTitleName( SELFKEYWORD, XPOS,YPOS, "resource name" )  
AttachCardTypeName( SELFKEYWORD, "resource name" )  
AttachCenter( SELFKEYWORD, LEVEL, PLAYER, "typename", DATA1, DATA2, DATA3, DATA4 )  
AttachCenterUpgrade( SELFKEYWORD, "nextupgrade\_cardname" )  
AttachColonyPlanet( SELFKEYWORD )  
AttachEncounterFight( SELFKEYWORD, STRENGTH )  
AttachFlightEnd( SELFKEYWORD, ENGINE )  
AttachForTargetPlanet( SELFKEYWORD, "card name" )  
AttachGfxElement(SELFKEYWORD,"gfxname\_resource",X\_POSITION,Y\_POSITION )  
AttachPayment( SELFKEYWORD, AMOUNT )  
AttachPaymentResource( SELFKEYWORD, "resource name", AMOUNT )  
AttachProduction( SELFKEYWORD, "resource name", DICEVALUE, AMOUNT )  
AttachResultResourceJoker( SELFKEYWORD, "result", "mode", AMOUNT, COST )  
AttachResultResource( SELFKEYWORD, "result", "type name", AMOUNT )  
AttachResultType( SELFKEYWORD, "result", "type name", AMOUNT )  
AttachResultReplaceCard( SELFKEYWORD, "result", "source card", "destination card" )  
AttachTrade( SELFKEYWORD, "resource name", "trade mode", COST )  
AttachTargetPlanetRequirement( SELFKEYWORD, "type", AMOUNT )  
AttachTargetPlanetRequirement\_Center( SELFKEYWORD, "center unique name" )  
AttachTargetPlanetRequirement\_Resource( SELFKEYWORD, "resource name", AMOUNT )  
AttachTradePlanet( SELFKEYWORD )  
AttachTransportDestination( SELFKEYWORD )  
AttachTypedGfx( SELFKEYWORD, "typename", "resource name" )

```
AttachVariableTrade( SELFKEYWORD, "resource name", "trade mode", COST, AMOUNT )  
SetCardDataTransport( SELFKEYWORD, AMOUNT )
```

## Befehlsliste

### AttachAddAction

#### FUNCTION

Addiert Werte für den Spieler hinzu, sobald sich diese Karte in den Handkartenstapeln oder Zentrumsstapeln des Spielers befinden. Dies gilt nicht für die Zentrumsauswahlstapel.  
Es wird diese Anzahl von Aktionen hinzugefügt, diese können in der Flugphase verwendet werden.  
Mehr als 9 Aktionen werden nicht gewertet.

Hinweis: Ein Zentrum II benötigt nur EINE Aktion, da die Zentrumskarte I noch aktiv im Stapel bleibt und ebenfalls eine Aktion hinzufügt.

#### SYNOPSIS

```
AttachAddAction( self, ACTION_NUM )
```

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.  
ACTION\_NUM: Anzahl Aktionen

#### DEMO

```
AttachAddAction( self, 1 )
```

### AttachAddStore

#### FUNCTION

Addiert Werte für den Spieler hinzu, sobald sich diese Karte in den Handkartenstapeln oder Zentrumsstapeln des Spielers befinden. Dies gilt nicht für die Zentrumsauswahlstapel.  
Es wird diese Anzahl von Laderaum für alle Lager hinzugefügt.  
Mehr als 9 Lagereinheiten werden nicht gewertet.

#### SYNOPSIS

```
AttachAddStore( SELFKEYWORD, STORE_NUM )
```

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.  
STORE\_NUM : Addiert diese Anzahl Lagerkapazität.

#### DEMO

```
AttachAddStore( self, 1 )
```

## **AttachBordStateComment**

### **FUNCTION**

Diese Funktion lässt den Roboter Bord einen Kommentar sprechen, sobald der entsprechende STATE (Programmteil) oder SUBSTATE (Unterprogrammteil) erreicht wird.  
Die Bord Kommentare sind nur an einer speziellen Karte angehängt, die auf den Aktions Stapel (action Stack) liegen muss.

### **SYNOPSIS**

AttachBordStateComment( SELFKEYWORD, "state", "substate", CALLED, TURN, "text", SPEECH\_TAKE )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"state" : Name des STATE, siehe Tabelle.

"substate" : Name des SUBSTATE, siehe Tabelle. "none" = jeder Unterschnitt.

CALLED : 0=in jedem Aufruf, sonst bei dem CALLED Aufruf des Teils

TURN : 0 = beginning, >0 = in dieser Runde soll der Kommentar kommen.

"text" : mname der Text Resource aus der Bibliothek.

SPEECH\_TAKE : 0 = keiner, >0 = Sprachausgabe des Takes zum Abspielen.

### **DEMO**

AttachBordStateComment( self, "flight", "none", 2, 0, "BORD\_SAYHELLOWOLRD", 120 )

## **AttachCardText**

### **FUNCTION**

Unterer Text der Karte, Aufgabentexte.

### **SYNOPSIS**

AttachCardText( SELFKEYWORD, "resource name" )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"resource name" : Name der Textresource der Bibliothek.

### **DEMO**

AttachCardText( SELFKEYWORD, "CT\_NEWSCENTER\_1\_1" )

## **AttachCardTitleName**



## FUNCTION

Überschrift der Karte, mittlerer Text.

## SYNOPSIS

AttachCardTitleName( SELFKEYWORD, XPOS,YPOS, "resource name" )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"resource name" : Name der Textresource der Bibliothek.

## DEMO

AttachCardTitleName( SELFKEYWORD, XPOS,YPOS, "resource name" )

## AttachCardTypeName

## FUNCTION

Linke obere Überschrift der Karte.

## SYNOPSIS

AttachCardTypeName( SELFKEYWORD, "resource name" )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"resource name" : Name der Textresource der Bibliothek.

## DEMO

AttachCardTypeName( SELFKEYWORD, self.name )

## AttachCenter

## FUNCTION

Erweitert diese Karte zu einer Zentrumskarte. Die Zentrumskarten sollten immer auf den Zentrumsstapeln liegen, nicht in der Flugreihe, den Aufgaben oder Sektoren.

Hinweis: Darf ausschließlich für Zenrtumskarten verwendet werden.

## SYNOPSIS

AttachCenter( SELFKEYWORD, LEVEL, PLAYER, "typename", DATA1, DATA2, DATA3, DATA4 )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

LEVEL : 1 oder 2 je nach Zentrum I oder Zentrum II

PLAYER : 0=für beide Spieler, 1 = Spieler 1, 2 = Spieler 2  
"typename" : Typ des Zentrums, siehe Liste im Anhang der Namen

DATA1 – DATA4: Je nach Name und Art des Zentrums belegte Variablen:

"trade" :

DATA1 = Anzahl zu handelnder Waren vom Mitspieler

DATA2 = Preis der Waren

"scan"

DATA1 = Scantiefe ( wieviel Karten können betrachtet werden )

"transport"

DATA1 = Anzahl zu transportierender Waren

## DEMO

AttachCenter( self, 1,2, "science" ,0 ,0 ,0 ,0)

### AttachCenterUpgrade

## FUNCTION

Macht diese Zentrumskarte zu einer anderen erweiterbar.

Hinweis: Arbeitet mit jedem Level des Zentrums.

## SYNOPSIS

AttachCenterUpgrade( SELFKEYWORD, "nextupgrade\_cardname" )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

## DEMO

AttachCenterUpgrade( self, "card.productioncenter\_2" )

### AttachColonyPlanet

## FUNCTION

Fügt an diese Karte die Funktionalität "Kolonieplanet" an. Der Planet kann als Kolonie in den Handkartenstapel übernommen werden. Siegpunkte werden nicht automatisch angefügt.

## SYNOPSIS

AttachColonyPlanet( SELFKEYWORD )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

## DEMO

AttachColonyPlanet( self )

## AttachEncounterFight

## FUNCTION

Definiert eine Karte als Piraten.

## SYNOPSIS

AttachEncounterFight( SELFKEYWORD, STRENGTH )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

STRENGTH: gibt die Stärke des Piraten an

## DEMO

AttachEncounterFight( self, 4 )

## AttachFlightEnd

## FUNCTION

Stoppt den aktuelle Flug wenn diese Karte in die Flugreihe kommt und der Spieler nicht über die angegebene Anzahl an Antrieben verfügt.

## SYNOPSIS

AttachFlightEnd( SELFKEYWORD, ENGINE )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

ENGINE : Anzahl Antriebe die mindestens verfügbar sein muss um fortzufahren.

## DEMO

AttachFlightEnd( self, 4 )

## AttachForTargetPlanet

## FUNCTION

Gibt bei einer Abenteuerkarte den Zielplaneten an, der dieser Aufgabe zugeordnet ist.

## SYNOPSIS

AttachForTargetPlanet( SELFKEYWORD, "card name" )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.  
"card name": Name des Zielplaneten

### DEMO

AttachForTargetPlanet( self, "CN\_PALLAS" )

## AttachGfxElement

### FUNCTION

Fügt ein grafisches Element an eine Karte an. Dieses wird durch einen Namen aus den Bibliotheken gesetzt und an dieser Position dargestellt. Sind die Koordinaten uasserhalb der Karte, wird dieses Element nicht sichtbar sein.

Hinweis: Aus den bestehenden Karten lassen sich in Verbindung mit den Listen die bestehenden Grafiken setzen.

### SYNOPSIS

AttachGfxElement(SELFKEYWORD,"gfxname\_resource",X\_POSITION,Y\_POSITION )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"gfxname\_resource" : Name der Resource.

X\_POSITION : Position auf der Karte, x Achse

Y\_POSITION : Position auf der Karte, y Achse

### DEMO

AttachGfxElement( self, "gfxelem.hades.winpoint",10,10 )

## AttachPayment

### FUNCTION

Bestimmt bei einer Aufgabenkarte, was der Spieler bezahlen muss um diese zu erfüllen. In diesem Falle muss der Spieler diese Anzahl Astro bezahlen.

### SYNOPSIS

AttachPayment( SELFKEYWORD, AMOUNT )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

AMOUNT : Astro

### DEMO

AttachPayment( self , 5 )

## **AttachPaymentResource**

### **FUNCTION**

Bestimmt bei einer Aufgabenkarte, was der Spieler bezahlen muss um diese zu erfüllen. In diesem Falle muss der Spieler diese bestimmte Resource in dieser Anzahl bezahlen.

### **SYNOPSIS**

AttachPaymentResource( SELFKEYWORD, "resource name", AMOUNT )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"resource name" : Name der Resource, siehe Tabelle

AMOUNT : Anzahl

### **DEMO**

AttachPaymentResource( self , "erz", 2 )

## **AttachProduction**

### **FUNCTION**

Fügt an diese Karte die Produktionsfunktion an. Bei jedem Würfelwurf auf der angegebenen Zahl wird diese Resource in die Produktionsliste aufgenommen und diese Anzahl produziert. Bei doppelter Produktion entscheidet der Spieler anhand der Auswahlliste.

### **SYNOPSIS**

AttachProduction( SELFKEYWORD, "resource name", DICEVALUE, AMOUNT )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"resource name" : Name der Resource, siehe Tabelle.

DICEVALUE : Würfelergebnis bei dem produziert wird (1,2,3)

AMOUNT : Anzahl die produziert wird.

### **DEMO**

AttachProduction( self , "erz", 2, 1 )

## **AttachResultResourceJoker**

### **FUNCTION**

Die Ergebnisfunktionen ("result") einer Karte werden nach einem erfolgreichen Kampf oder einer erfolgreich ausgeführten Aufgabe aufgerufen. Sie bestimmen eine Aktion oder ein Ergebnis das ausgeführt wird.

Diese Funktion ruft ein Auswahlmenu auf in dem der Spieler sich diese Rohstoffe aussuchen kann. Je nach Modi bekommt er diese umsonst, muss dafür bezahlen oder er muss sie abgeben.

Hinweis: Es kann nur eine Joker Funktion gelistet sein !

## SYNOPSIS

AttachResultResourceJoker( SELFKEYWORD, "result", "mode", AMOUNT, COST )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"result" : "won" "lost"

"mode" : "buy" "sell" "getfree" "mustdrop"

AMOUNT : Anzahl

COST :

## DEMO

AttachResultResourceJoker( self, "won", "getfree", 2, 0 )

## AttachResultResource

## FUNCTION

Die Ergebnisfunktionen ("result") einer Karte werden nach einem Kampf oder einer erfolgreich ausgeführten Aufgabe aufgerufen. Sie bestimmen eine Aktion oder ein Ergebnis das ausgeführt wird. Es können mehrere "Result" Funktionen gelistet sein.

Diese Funktion liefert eine Anzahl Ressourcen an den Spieler für den Fall "Gewonnen" oder bei "Verloren" werden diese abgezogen.

## SYNOPSIS

AttachResultResource( SELFKEYWORD, "result", "type name", AMOUNT )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"won" "lost"

AMOUNT : Anzahl

## DEMO

AttachResultResource( self, "won", "erz", 2 )

## AttachResultType

## FUNCTION

Die Ergebnisfunktionen ("result") einer Karte werden nach einem Kampf oder einer erfolgreich ausgeführten Aufgabe aufgerufen. Sie bestimmen eine Aktion oder ein Ergebnis das ausgeführt wird. Es können mehrere "Result" Funktionen gelistet sein.

Diese Funktion liefert eine Anzahl eines Typs an den Spieler für den Fall "Gewonnen" oder bei "Verloren" werden diese abgezogen, bzw. zerstört.

Hinweis: Bei center kann nur eine "Type" Funktion aufgerufen werden.

## SYNOPSIS

AttachResultType( SELFKEYWORD, "result", "type name", AMOUNT )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"result" : "won" "lost"

"type name" : "astro", "engine", "weapon", "flightend", "center"

AMOUNT : Anzahl

## DEMO

AttachResultType( self, "won", "engine",1 )

## AttachResultReplaceCard

## FUNCTION

Die Ergebnisfunktionen ("result") einer Karte werden nach einem Kampf oder einer erfolgreich ausgeführten Aufgabe aufgerufen. Sie bestimmen eine Aktion oder ein Ergebnis das ausgeführt wird. Es können mehrere "Result" Funktionen gelistet sein.

Diese Funktion ersetzt beliebige Karten durch andere.

## SYNOPSIS

AttachResultReplaceCard( SELFKEYWORD, "result", "source card", "destination card" )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"result" : "won" "lost"

"source card" : Name der zu ersetzenden Karte

"destination card" : Name der neuen Karte

## DEMO

AttachResultReplaceCard( self )

## **AttachTrade**

### **FUNCTION**

Fügt die Handelsfunktion an diese Karte an. Die Karte wird zur vollständigen Handelskarte innerhalb des Fluges oder als Planetenhandel in der Bauphase.

Hinweis: Nur die ERSTE gefundene Handelsfunktion wird ausgewertet sollten mehr als eine innerhalb der Karte vorhanden sein.

### **SYNOPSIS**

AttachTrade( SELFKEYWORD, "resource name", "trade mode", COST )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"resource name" : Name der Resource für Handel, siehe Tabelle.

"trade mode" : Modus in dem gehandelt werden kann: "buy", "sell", "buysell"

AMOUNT : Anzahl die gehandelt werden kann.

### **DEMO**

AttachTrade( self, "carbon", "buy", 2 )

## **AttachTargetPlanetRequirement**

### **FUNCTION**

Fügt ein "requirement" an eine Aufgaben (task) Karte an. Dieses wird geprüft bevor die Karte freigegeben wird, zur Lösung durch den Spieler. Der Spieler benötigt eine bestimmte Anzahl des angegebenen Typs um die Aufgabe lösen zu können.

KI: Nur EIN "requirement" pro Karte wird berücksichtigt und brechnet. Für ein KI Spiel sind einfache 1-Zentrums Anforderungen aufzubauen.

### **SYNOPSIS**

AttachTargetPlanetRequirement( SELFKEYWORD, "type", AMOUNT )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"type": Gibt an, was zum lösen der Aufgabe benötigt wird

"astro"

"engine"

"weapon"

"siegpunkt"

"freundschaft"

"orden"

AMOUNT: Diese Anzahl des angegebenen Typs wird benötigt



## DEMO

AttachTargetPlanetRequirement( self, "engine", 4 )

### **AttachTargetPlanetRequirement\_Center**

## FUNCTION

Fügt ein "requirement" an eine Aufgaben (task) Karte an. Dieses wird geprüft bevor die Karte freigegeben wird, zur Lösung durch den Spieler.

Der Spieler benötigt eine Zentrum dieser Art um die Aufgabe lösen zu können.

KI: Nur EIN "requirement" pro Karte wird berücksichtigt und brechnet. Für ein KI Spiel sind einfache 1-Zentrums Anforderungen aufzubauen.

## SYNOPSIS

AttachTargetPlanetRequirement\_Center( SELFKEYWORD, "center unique name" )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"center unique name" : Name des Zentrums, siehe Liste der Zentren.

## DEMO

AttachTargetPlanetRequirement\_Center( self, "transport" )

### **AttachTargetPlanetRequirement\_Resource**

## FUNCTION

Mit dieser Funktion wird für eine Aufgabenkarte bestimmt, welche Ressource zum lösen der Aufgabe benötigt wird.

HINWEIS:

Der Computergegner berechnet nur das erste Auftreten dieser Funktion in der Kartendefinition und wird weitere Funktionen dieses Typs ignorieren.

## SYNOPSIS

AttachTargetPlanetRequirement\_Resource( SELFKEYWORD, "resource name", AMOUNT )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"resourcenname": name der benötigten Ressource

AMOUNT: Anzahl der Ressource, die zum lösen benötigt wird

## DEMO

AttachTargetPlanetRequirement\_Resource( self, "carbon", 2 )

## **AttachTradePlanet**

### **FUNCTION**

Fügt an diese Karte die Funktionalität "Handelsplanet" an. Der Planet kann als Handelsstation in den Handkartenstapel übernommen werden. Freundschaftspunkte werden nicht automatisch angefügt.

### **SYNOPSIS**

AttachTradePlanet( SELFKEYWORD )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

### **DEMO**

AttachTradePlanet( self )

## **AttachTransportDestination**

### **FUNCTION**

Markiert diesen Planeten als Ziel für eine Transportaufgabe. Unabhängig von der Aufgabe ist die Transportfunktion aktiv sobald der Planet angefliegen wird.

### **SYNOPSIS**

AttachTransportDestination( SELFKEYWORD )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

### **DEMO**

AttachTransportDestination( self )

## **AttachTypedGfx**

### **FUNCTION**

Mit dieser Funktion werden einem Zentrum die verschiedenen Animationen zugewiesen. Diese Funktion wird nur bei Zentrumskarten eingesetzt.

### **SYNOPSIS**

AttachTypedGfx( SELFKEYWORD, "typename", "resource name" )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

"typename": gibt an, welche Animation mit dem Befehl definiert werden soll  
"center\_still"  
"center\_idle"  
"center\_buildvideo"  
"center\_upgradevideo"  
"center\_downgradevideo"  
"resourcenname": der Ressourcenname des Videos oder der Grafik, die auf dem Zentrum für die gewählte Aktion gezeigt werden soll.

## DEMO

```
AttachTypedGfx( self, "center_idle", "ship.video.center.trade.one.idle" )
```

## AttachVariableTrade

### FUNCTION

Definiert eine Karte als Handelsplaneten mit variablem Handel. Diese Art der Handelskarte ist flexibler als ein normaler Handelsplanet. Alle normalen "AttachTrade" Funktionen werden für diese Karte ignoriert, sobald diese Funktion benutzt wird.

### SYNOPSIS

```
AttachVariableTrade( SELFKEYWORD, "resource name", "trade mode", COST, AMOUNT )
```

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.  
resource name: gibt an, welche Ressource hier gehandelt wird.  
"any" - Sonderfall, jede Ressource kann gehandelt werden.  
trademode: gibt an, welche Handelsaktionen möglich sind  
"buy" - es kann nur gekauft werden  
"sell" - es kann nur verkauft werden  
"buysell" - kaufen und verkaufen sind beide möglich  
COST: definiert den Preis, zu dem die Ressourcen angeboten werden  
AMOUNT: gibt die maximale Anzahl an Handelsaktionen pro Flug an

## DEMO

```
AttachVariableTrade( self, "any", "sell", 4, 2 )
```

## SetCardDataTransport

### FUNCTION

Greift auf den Wert des internen Kartenspeichers zu und setzt das Feld "transport" auf den angegebenen Wert. Es sind damit Transport mit der Anzahl Waren möglich.

## **SYNOPSIS**

SetCardDataTransport( SELFKEYWORD, AMOUNT )

SELFKEYWORD: "self", weist auf das Kartenobjekt hin.

AMOUNT : Anzahl der möglichen zu transportierenden Waren insgesamt.

## **DEMO**

SetCardDataTransport( self, 4 )

## **Hinweise / Legal notes**

Sternenschiff Catan setzt für das Scripting folgende Technologien ein:

Lua - An Extensible Extension Language

Copyright (C) 2002 Tecgraf, PUC-Rio. All rights reserved.

Homepage: <http://www.lua.org>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

# Inhalt

Impressum.....	2
Installation.....	3
Warnung.....	4
Starten des Spiels .....	4
Scripting.....	5
Interpretation.....	5
Script Ausführung.....	5
Regeln zum Entwurf der Karten.....	6
<b>Kartendefinition.....</b>	<b>6</b>
<b>Kartendesign.....</b>	<b>7</b>
<b>Kartenwerte.....</b>	<b>7</b>
<b>Kartenfunktionen.....</b>	<b>8</b>
<b>Interne Kartenfunktionen.....</b>	<b>9</b>
ANHANG 1: Tabellen.....	10
<b>Stapel (stacks).....</b>	<b>10</b>
<b>Erläuterung Stapel Tabelle.....</b>	<b>11</b>
<b>STATES.....</b>	<b>12</b>
<b>SUBSTATES.....</b>	<b>13</b>
<b>Zentren (center type names).....</b>	<b>14</b>
<b>Rohstoffe (resource names).....</b>	<b>14</b>
ANHANG 2: Befehlsliste, Script Befehle.....	15
<b>Übersicht.....</b>	<b>15</b>
<b>Befehlsliste.....</b>	<b>16</b>
ANHANG 3: Befehlsliste, Objekt Befehle.....	25
<b>Übersicht.....</b>	<b>25</b>
<b>Befehlsliste.....</b>	<b>25</b>
Hinweise / Legal notes.....	39