ArkhamHorror Skript

# Allgemeines

## Funktion

AHS ist eine Skriptsprache für ArkhamHorror zur Definition weiterer Spiel-Objekte, wie Waffen, Verbündete, Begegnungen, etc.

Es ist ein Generator für ArkhamHorror’s JavaScript, das vom ArkhamHorror Server verwendet wird. Die Dateien werden intern in JS übersetzt und dann an vom Server verwendet.

Es bietet eine Vereinfachung gegenüber dem Original JS Code.

## Genereller Aufbau

In AHS werden Objekte von bestimmten Klassen definiert. Die verfügbaren Klassen sind fest definiert, und bestimmen, wie mit dem Objekt umgegangen wird. Jedes Objekt hat einen eindeutigen Namen.

Objekte mancher Klassen können mehrfach im Spiel eingefügt werden (z.B. Gegenstände). Hier wird die Multiplizität nach dem Objektnamen angegeben, durch Doppelpunkt (:) getrennt.

Die Eigenschaften der Objekte werden über Attribute definiert. Jede Klasse legt fest, welche Attribute verfügbar sind, und welche davon verpflichtend sind. Die Attribute werden durch geschwungene Klammern ({ und }) umschlossen.

Attribute haben einen bestimmten Typ. Dieser bestimmt, wie die Werte angegeben werden müssen. Für mache Attribute sind mehrere alternative Typen möglich. Die Interpretation mancher Typen wird durch das Attribut festgelegt.

Attribute haben einen Namen, der mit Doppelpunkt (:) vom Wert getrennt wird. Attributwerte werden mit einem Semikolon (;) abgeschlossen.

Namen haben folgendes Format: Groß-/Kleinbuchstaben, Underscore (\_), Ziffern; dürfen nicht mit Ziffer beginnen.

Manche Attribute werden automatisch vom System vergeben. Diese dürfen nicht angegeben werden.

Bsp.:

CommonItem Food : 2 {  
 Attribute: Value;  
}

## JS-Übersetzung

Die Übersetzung von AHS in JS Code werden die AHS-Klassen in JS-Objekte umgewandelt. Dabei wird aus jedem AHS-Attribut eine Property des Objekts. Weitere Properties können durch AHS automatisch eingefügt werden (sind in Kapitel 3 als „automatisch“ gekennzeichnet).

Für jede AHS-Klasse gibt es eine klassenspezifische createXXX und registerXXX JS-Methode, im globalen Objekt game, die aufgerufen werden. Gewisse Register-Funktionen benötigen auch eine Anzahl, die wird automatisch angegeben (durch Multiplizität in AHS).

Für einmalige Objekte wird eine eindeutige ID vergeben (in Kapitel 3 als „ID“ gekennzeichnet), für mehrfache Objekte eine eindeutige Typ-ID (in Kapitel 3 als „TID“ gekennzeichnet). Diese IDs / TIDs setzen sich aus einem Klassen-Typ spezifischen Präfix, und dem Klassennamen zusammen (z.B. "CI\_Dynamite").

Zusätzlich wird für jede AHS-Klasse eine Konstante mit registerConstant erzeug, sodass im JS-Code mit {KlassenTyp}.{KlassenName} darauf referenziert werden kann (z.B. CommonItem.Dynamite).

# Attribut Typen

## Kommentar

Kommentare werden ignoriert. Es gibt Zeilenkommentare, startend mit 2 Schrägstrichen (//), die bis zum Zeilenende laufen, und Blockkommentare, die von Schrägstrich-Stern (/\*) bis Stern-Schrägstrich (\*/)laufen (auch über mehrere Zeilen). Bsp.:

Das wird verwendet // Das hier wird ignoriert (Zeilenkommentar)  
Das nicht /\* Das schon  
und das auch (Blockkommentar) \*/ das nicht mehr

## Boolean

Boolscher Wert. Kann nur die 2 Werte „true“ und „false“ annehmen.

## Ganzzahl (Number)

Zahl. Folge von Ziffern (0-9). Sollte nicht mit „0“ beginnen.

## String

Text. Wird entweder mit Doppelten (") oder Einfachen (') Anführungsstrichen umschlossen.

Wenn in einem String ein Anführungsstrich verwendet wird, muss er mit Backslash (\) escaped werden (z.B. "Test \"mit\" 'Anführungsstrichen'").

Mehrere Strings können per Plus (+) zusammengesetzt werden. Das ermöglicht das Trennen des Strings auf mehrere Zeilen zur besseren Lesbarkeit im Code. Es können Strings mit einfachen und doppelten Anführungsstrichen zusammengesetzt werden. Bsp.:

"Dieser Text ist zu "+  
"lang, um ihn gut in "+  
'einer Zeile im "Code" zu lesen'

## Name

Namen sind Anzeigenamen von Objekten vom Typ String. Wenn kein Name angegeben wird, wird der Objekt-Name verwendet, wobei CamelCase in Leerzeichen umgewandelt wird.

Bsp.:

AncientTome 🡺 Name = "Ancient Tome"

## Aufzählungswert (Enum)

Für gewisse Attribute sind Attribut-spezifische Aufzählungswerte aus einer Liste möglich (z.B. Bewegungstyp bei Monstern). Bei diesen wird einfach der Wert angegeben.

Bsp.:

movementType: Flying;

### JS-Übersetzung

Jedes Enum-Attribut hat einen gewissen Typ (in Kapitel 3 angegeben, Werte in Kapitel 4 gelistet). Für diese Werte gibt es JS-Konstanten, im globalen Objekt Constants.{Type}. Dieser Präfix wird Attribut-spezifisch bei der Übersetzung eingefügt.

## Referenzen (IDRef)

Manche Attribute sind Beziehungen zu anderen Objekten (z.B. Spezialfertigkeiten bei Investigatoren). Hier wird die Objektklasse und der Objektname durch Punkt (.) getrennt angegeben.

Bsp.:

specialAbilityId: Object.JennyBarnesSpecialAbility;

### JS-Übersetzung

ID-Referenzen werden als Strings ausgegeben, indem der Klassen-spezifische Präfix und Name zusammengesetzt wird (vgl. 1.3).

## Listen (Array)

Gewisse Attribute erlauben Listen von Aufzählungswerten oder Referenzen. Diese werden von runden Klammern umschlossen (( und )) und durch Komma (,) getrennt.

Bsp.:

attributes: ( Unending, Physical\_Resistance );

## Komplex (Complex)

Ein Komplex hat eine Attribut-Spezifische Bedeutung (siehe Attribute unten). Er wird mit eckigen Klammern ([ und ]) umschlossen. Bsp.:

[  
 // Komplexer Wert. Bedeutung Attribut-Abhängig  
]

## Funktion (Function)

Manche Attribute sind Funktionen (siehe Attribute unten, z.B. onMove bei Monstern). Funktionen sind zu komplex, um in AHS abzubilden. Funktionen werden daher direkt in JS Code angegeben.

Eine AHS-Funktion wird mit 3 geschwungenen Klammern umschlossen ({{{ und }}}).

Bsp.:

onMove: {{{  
 // JS-Code  
}}};

## Literale (Literal)

Manche Attributwerte sind zu komplex, um sie in AHS abzubilden. Dafür werden Literale verwendet, die 1:1 in den JS-Code übernommen werden. Literale werden mit 3 spitzen Klammern umschlossen (<<< und >>>).

Literale können immer bei Attributwerten verwendet werden.

Bsp.:

attribute: <<< /\* JS-Code hier \*/ >>>;

# Klassen

## Action

### Beschreibung

Eine Aktion, die im Spiel etwas verändert. Reine Aktionen werden automatisch durchgeführt. Ein Beispiel ist die Spezialfertigkeit von Jenny Barnes: Sie bekommt in jeder Upkeep Phase automatisch $1.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| id | Automatisch | ID | Eindeutige ID |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibugnstext |
| phases | Verpflichtend | Array(Enum): GamePhases | Spielphasen, in denen die Aktion durchgeführt werden kann |
| activate | Verpflichtend | Function | Aktion der Action |

### JS-Übersetzung:

Create: game.createAction(obj)  
Register: game.registerAction(action)  
ID-Prefix: AC  
Constant-Scope: Action

## Option

### Beschreibung

Eine Spieloption, die dem Spieler angeboten wird. Eine Option hat eine Referenz auf eine Action, die bei Aktivierung durchgeführt wird. Optionen können mit Kosten verbunden sein, oder nur unter gewissen Umständen verfügbar sein. Die Beschreibung wird von der referenzierten Aktion übernommen.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| Id | Automatisch | ID | Eindeutige ID |
| Name | Optional | Name | Name (Anzeige) |
| isAvailable | Optional | Function | Prüft, ob die Option verfügbar ist. Muss Boolean zurückliefern. Standard: true (immer verfügbar) |
| actionId | Verpflichtend | IDRef: Action | Referenz zur Action |
| continueType | Optional | Enum: ContinueType | Ob der Zug nach der Option beendet ist, oder eine weitere Option gewählt werden kann |
| chooseType | Optional | Enum: ChooseType | Wie die Option gewählt werden kann in Kombination mit anderen Optionen |
| Costs | Optional | Complex: Costs | Kosten der Option |

### JS-Übersetzung:

Create: game.createOption(obj)  
Register: game.registerOption(option)  
ID-Prefix: OP  
Constant-Scope: Option

## QuickOption

### Beschreibung

Kombination von Option und Action. Aus den angegebenen Attributen wird eine Action und eine Option dafür erstellt.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| id | Automatisch | ID | Eindeutige ID |
| name | Optional | Name | Name (Anzeige) |
| isAvailable | Optional | Function | Siehe Option |
| continueType | Optional | Enum: ContinueType | Siehe Option |
| chooseType | Optional | Enum: ChooseType | Siehe Option |
| costs | Optional | Complex: Costs | Siehe Option |
| description | Optional | String | Siehe Action |
| phases | Verpflichtend | Array(Enum): GamePhases | Siehe Action |
| activate | Verpflichtend | Function | Siehe Action |

### JS-Übersetzung:

Create: game.quickOption(obj)  
Register: Nicht verfügbar. quickOption registriert die Option automatisch  
ID-Prefix: OP  
Constant-Scope: Option

## FieldOption

### Beschreibung

Option auf einem Spielfeld. Das sind die speziellen Begegnungen in gewissen Feldern (z.B. Heilung in St. Mary’s Hospital). Es entspricht einer QuickOption, plus Feld

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| id | Automatisch | ID | Eindeutige ID |
| name | Optional | Name | Name (Anzeige) |
| isAvailable | Optional | Function | Siehe Option |
| continueType | Optional | Enum: ContinueType | Siehe Option |
| chooseType | Optional | Enum: ChooseType | Siehe Option |
| costs | Optional | Complex: Costs | Siehe Option |
| description | Optional | String | Siehe Action |
| phases | Verpflichtend | Array(Enum): GamePhases | Siehe Action |
| activate | Verpflichtend | Function | Siehe Action |
| field | Verpflichtend | Enum: Field | Feld, auf dem die Option angeboten wird |

### JS-Übersetzung:

Create: game.quickOption(obj)  
Register: addFieldOtion(Field, objId)  
ID-Prefix: OP  
Constant-Scope: Option

## CommonItem

### Beschreibung

Profane Gegenstände, wie Waffen, Bücher, Nahrung, etc. CommonItems können Mehrfach vorkommen.

CommonItems werden in den Stapel für CommonItems gelegt, und können im Spiel gezogen werden.

CommItems können Modifikatoren haben (z.B. Waffen geben einen Physischen Schaden), oder Optionen zur Verfügung stellen (z.B. Buch lesen).

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| tid | Automatisch | TID | Eindeutige Typ ID |
| type | Automatisch | - | Interner Typ |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibung |
| hands | Optional | Number | Anzahl benötigter Hände (Waffen) |
| price | Optional | Number | Preis beim Kauf |
| exhaustable | Optional | Boolean | Ob das Objekt nach der Benutzung getappt wird, und erst in der nächsten Runde verfügbar ist (Standard: false) |
| optionIds | Optional | Array(IDRef): Option | Liste der Optionen, die das Objekt zur Verfügung stellt |
| attributes | Optional | Array(Enum): ObjectAttributes | Objekt-Attribute |
| modifications | Optional | Complex (Mods) | Modifikatoren (v.a. Waffen) |
| onAddToInventory | Optional | Function | Funktion, wenn das Objekt erhalten wird |
| onRemoveFromInventory | Optional | Function | Funktion, wenn das Objekt verloren wird |
| properties | Optional | Literal | Dynamische Properties (fortgeschritten) |

### JS-Übersetzung:

Create: game.createObject(obj)  
Register: game.registerMultiObject(count, object)  
ID-Prefix: CI  
Constant-Scope: CommonItem  
Type: CommonItem

## UniqueItem

### Beschreibung

Magische Gegenstände, wie Waffen, magische Steine, , etc. UniqueItems können Mehrfach vorkommen.

UniqueItems werden in den Stapel für UniqueItems gelegt, und können im Spiel gezogen werden.

UniqueItems können Modifikatoren haben (z.B. Waffen geben einen Magischen Schaden), oder Optionen zur Verfügung stellen (z.B. Stein verwenden).

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| tid | Automatisch | TID | Eindeutige Typ ID |
| type | Automatisch | - | Interner Typ |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibung |
| hands | Optional | Number | Anzahl benötigter Hände (Waffen) |
| price | Optional | Number | Preis beim Kauf |
| exhaustable | Optional | Boolean | Ob das Objekt nach der Benutzung getappt wird, und erst in der nächsten Runde verfügbar ist (Standard: false) |
| optionIds | Optional | Array(IDRef): Option | Liste der Optionen, die das Objekt zur Verfügung stellt |
| attributes | Optional | Array(Enum): ObjectAttributes | Objekt-Attribute |
| modifications | Optional | Complex (Mods) | Modifikatoren (v.a. Waffen) |
| onAddToInventory | Optional | Function | Funktion, wenn das Objekt erhalten wird |
| onRemoveFromInventory | Optional | Function | Funktion, wenn das Objekt verloren wird |
| properties | Optional | Literal | Dynamische Properties (fortgeschritten) |

### JS-Übersetzung:

Create: game.createObject(obj)  
Register: game.registerMultiObject(count, object)  
ID-Prefix: UI  
Constant-Scope: UniqueItem  
Type: UniqueItem

## Skill

### Beschreibung

Fertigkeiten. Skills können Mehrfach vorkommen.

Skills werden in den Stapel für Skills gelegt, und können im Spiel gezogen werden.

Skills haben Modifikatoren, die die Attribute des Charakters ändern (Luck, Lore, Fight, Will, Speed, Sneak), oder weitere Würfel auf Attribut-Tests geben

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| tid | Automatisch | TID | Eindeutige Typ ID |
| type | Automatisch | - | Interner Typ |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibung |
| optionIds | Optional | Array(IDRef): Option | Liste der Optionen, die das Objekt zur Verfügung stellt |
| modifications | Optional | Complex (Mods) | Modifikatoren (v.a. Waffen) |
| onAddToInventory | Optional | Function | Funktion, wenn das Objekt erhalten wird |
| onRemoveFromInventory | Optional | Function | Funktion, wenn das Objekt verloren wird |
| properties | Optional | Literal | Dynamische Properties (fortgeschritten) |

### JS-Übersetzung:

Create: game.createObject(obj)  
Register: game.registerMultiObject(count, object)  
ID-Prefix: SK  
Constant-Scope: Skill  
Type: Skill

## Ally

### Beschreibung

Verbündete. Allies können nur einfach vorkommen.

Allies werden in den Stapel für Allies gelegt, und können im Spiel gezogen werden.

Allies können Modifikatoren haben (z.B. Ignorieren von Monster-Attributen), oder Optionen zur Verfügung stellen.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| tid | Automatisch | TID | Eindeutige Typ ID |
| type | Automatisch | - | Interner Typ |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibung |
| optionIds | Optional | Array(IDRef): Option | Liste der Optionen, die das Objekt zur Verfügung stellt |
| modifications | Optional | Complex (Mods) | Modifikatoren (v.a. Waffen) |
| onAddToInventory | Optional | Function | Funktion, wenn das Objekt erhalten wird |
| onRemoveFromInventory | Optional | Function | Funktion, wenn das Objekt verloren wird |
| properties | Optional | Literal | Dynamische Properties (fortgeschritten) |

### JS-Übersetzung:

Create: game.createObject(obj)  
Register: game.registerSingleObject(object)  
ID-Prefix: AL  
Constant-Scope: Ally  
Type: Ally

## Spell

### Beschreibung

Zaubersprüche. Spells können mehrfach vorkommen.

Spells werden in den Stapel für Spells gelegt, und können im Spiel gezogen werden.

Spells müssen durch eine Würfelprobe aktiviert werden. Es gibt Spells, die nur in bestimmten Situationen verfügbar sind und dann eine Option bieten (z.B. Find Gate), oder bieten einen Kampfbonus.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| tid | Automatisch | TID | Eindeutige Typ ID |
| type | Automatisch | - | Interner Typ |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibung |
| exhaustable | Optional | Boolean | Ob das Objekt nach der Benutzung getappt wird, und erst in der nächsten Runde verfügbar ist (Standard: false) |
| castTarget | Optional | Number | Zielwert bei Würfelprobe (Sandard 1) |
| castCost | Verpflichtend | Number | Sanity-Kosten zur Aktivierung des Spells |
| castAdjustment | Verpflichtend | Number | Anpassung des Lore-Wertes bei Würfelprobe |
| cast | Optional | Function | Spell Funktion |
| optionIds | Optional | Array(IDRef) | Liste der Optionen, die das Objekt zur Verfügung stellt (Sollte leer sein) |
| modifications | Optional | Complex (Mods) | Modifikatoren (v.a. Waffen) |
| onAddToInventory | Optional | Function | Funktion, wenn das Objekt erhalten wird |
| onRemoveFromInventory | Optional | Function | Funktion, wenn das Objekt verloren wird |
| properties | Optional | Literal | Dynamische Properties (fortgeschritten) |

### JS-Übersetzung:

Create: game.createObject(obj)  
Register: game.registerMultiObject(count, object)  
ID-Prefix: SP  
Constant-Scope: Spell  
Type: Spell

## BlessingCurse

### Beschreibung

Segnung / Fluch. Können mehrfach vorkommen.

Segnung / Fluch sind nicht direkt verfügbar, und können nur durch spezielle Aktionen erhalten werden (z.B Begegnungen, in der Kirche).

Es sollte nur eine Blessing und einen Curse geben, die den Success-Wert für Würfel anpasst.

Ein Spieler kann immer nur 1 Blessing ODER 1 Curse haben.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| tid | Automatisch | TID | Eindeutige Typ ID |
| type | Automatisch | - | Interner Typ |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibung |
| optionIds | Optional | Array(IDRef) | Liste der Optionen, die das Objekt zur Verfügung stellt |
| modifications | Optional | Complex (Mods) | Modifikatoren |
| onAddToInventory | Optional | Function | Funktion, wenn das Objekt erhalten wird |
| onRemoveFromInventory | Optional | Function | Funktion, wenn das Objekt verloren wird |
| properties | Optional | Literal | Dynamische Properties (fortgeschritten) |

### JS-Übersetzung:

Create: game.createObject(obj)  
Register: game.registerMultiObject(count, object)  
ID-Prefix: BC  
Constant-Scope: BlessingCurse  
Type: BlessingCurse

## SpecialObject

### Beschreibung

Spezielle Objekte, können mehrfach vorkommen.

Spezielle Objekte sind nicht direkt verfügbar, und können nur durch spezielle Aktionen erhalten werden (z.B. Hilfssherif-Marke, Logen-Mitgliedschaft, Bankanleihe, etc.).

Spezielle Objekte können Modifikatoren haben, oder Optionen und auch Aktionen zur Verfügung stellen. Aktionen werden automatisch ausgeführt, sind also z.B. bei der Bankanleihe in der Upkeep-Phase notwendig, sodass der Spieler automatisch würfeln muss, ob er die Bankanleihe verliert.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| tid | Automatisch | TID | Eindeutige Typ ID |
| type | Automatisch | - | Interner Typ |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibung |
| exhaustable | Optional | Boolean | Ob das Objekt nach der Benutzung getappt wird, und erst in der nächsten Runde verfügbar ist (Standard: false) |
| actionIds | Optional | Array(IDRef): Action | Liste der Aktionen, die das Objekt zur Verfügung stellt |
| optionIds | Optional | Array(IDRef): Option | Liste der Optionen, die das Objekt zur Verfügung stellt |
| modifications | Optional | Complex (Mods) | Modifikatoren (v.a. Waffen) |
| onAddToInventory | Optional | Function | Funktion, wenn das Objekt erhalten wird |
| onRemoveFromInventory | Optional | Function | Funktion, wenn das Objekt verloren wird |
| properties | Optional | Literal | Dynamische Properties (fortgeschritten) |

### JS-Übersetzung:

Create: game.createObject(obj)  
Register: game.registerMultiObject(count, object)  
ID-Prefix: OBJ  
Constant-Scope: SpecialObject  
Type: Special

## SpecialAbility

### Beschreibung

Spezielle Fähigkeiten von Investigatoren, können nur einfach vorkommen.

Spezielle Fähigkeiten können Modifikatoren haben, oder Optionen und auch Aktionen zur Verfügung stellen. Aktionen werden automatisch ausgeführt, sind also z.B. bei der Trust Fund (Jenny Barnse) in der Upkeep-Phase notwendig, sodass der Spieler automatisch $1 erhält.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| tid | Automatisch | TID | Eindeutige Typ ID |
| type | Automatisch | - | Interner Typ |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibung |
| exhaustable | Optional | Boolean | Ob das Objekt nach der Benutzung getappt wird, und erst in der nächsten Runde verfügbar ist (Standard: false) |
| actionIds | Optional | Array(IDRef): Action | Liste der Aktionen, die das Objekt zur Verfügung stellt |
| optionIds | Optional | Array(IDRef): Option | Liste der Optionen, die das Objekt zur Verfügung stellt |
| modifications | Optional | Complex (Mods) | Modifikatoren (v.a. Waffen) |
| onAddToInventory | Optional | Function | Funktion, wenn das Objekt erhalten wird |
| onRemoveFromInventory | Optional | Function | Funktion, wenn das Objekt verloren wird |
| properties | Optional | Literal | Dynamische Properties (fortgeschritten) |

### JS-Übersetzung:

Create: game.createObject(obj)  
Register: game.registerSingleObject(object)  
ID-Prefix: SA  
Constant-Scope: SpecialAbility  
Type: Special

## Monster

### Beschreibung

Ein Monster. Monster können mehrfach vorkommen.

Monster werden in den Monsterpool gelegt und bei Bedarf gezogen.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| tid | Automatisch | TID | Eindeutige Typ ID |
| name | Optional | Name | Name |
| description | Optional | String | Beschreibung |
| mythText | Optional | String | Weiterer Beschreibungstext. |
| movementType | Verpflichtend | Enum(Movement) | Art der Fortbewegung |
| awareness | Verpflichtend | Number | Wachsamkeit. Für Evade/Flight Tests |
| horrorAdjustment | Verpflichtend | Number | Anpassung für Horror-Probe |
| horrorDamage | Verpflichtend | Number | Horror-Schaden |
| combatAdjustment | Verpflichtend | Number | Anpassung für Kampf-Probe |
| combatDamage | Verpflichtend | Number | Kampf-Schaden |
| toughness | Verpflichtend | Number | Hitpoints |
| dimension | Optional | Enum(Dimension) | Heim-Dimension. Wenn leer wird eine zufällige Dimension vergeben |
| attributes | Optional | Array(Enum): MonsterAttributes | Attribute |
| modifications | Optional | Complex: Mods oder Function | Dynamische Modifikatoren |
| onMove | Optional | Function | Bewegung für Spezielle Bewegung |
| onDamage | Optional | Function(type, chr) | Funktion wenn das Monster einen Charakter angreift |
| onDefeat | Optional | Function | Funktion, wenn das Monster besiegt wurde |

### Anmerkungen

Monster mit movementType = Special müssen eine onMove Funktion haben.

onDamage wird aufgerufen, wenn Horror oder Combat-Schaden verursacht wurde. Der Parameter type gibt die Art des Schadens an (Enum DamageType: Horror, Combat, Nightmare, Overwhelm). Wenn die Funktion false zurückliefert, ist der Kampf automatisch beendet. Wenn die Funktion nichts oder true zurückliefert, wird der Kampf nach Standardregeln fortgesetzt.

onDefeat wird aufgerufen, wenn ein Spieler das Monster besiegt hat. Wenn die Funktion false zurückliefert, wird die das Monster nicht als Monstertrophäe an den Spieler gegeben (bzw. in den Monster-Pool zurückgelegt bei Unending Monstern). Wenn die Funktion true oder nichts zurückliefert, wird das getan (=Standardverhalten)

### JS-Übersetzung:

Create: game.createMonster (obj)  
Register: game.registerMonster(count, object)  
ID-Prefix: MO  
Constant-Scope: Monster

## ArkhamEncounter

### Bescheibung

Begegnung in Arkham. Eine Begegnung wird auf ein Feld gelegt, und bietet Optionen.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| description | Verpflichtend | String | Beschreibungstext |
| field | Verpflichtend | Enum: Field | Feld |
| options | Verpflichtend | Array(IDRef): Option | Optionen |

### Anmerkungen

Die Optionen sollten als QuickOption angelegt werden. Als phases muss ArkhamEncountery angegeben werden.

Feld muss ein Interior Feld sein.

### JS-Übersetzung:

Create: game.createArkhamEncounter (obj)  
Register: game.registerArkhamEncounter (object)  
ID-Prefix: AE  
Constant-Scope: ArkhamEncounter

## OtherWorldEncounter

### Bescheibung

Begegnung in Arkham. Eine Begegnung wird auf ein Feld gelegt, und bietet Optionen.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| description | Verpflichtend | String | Beschreibungstext |
| field | Optional | Enum: Field | Feld |
| color | Verpflichtend | Enum: OtherWorld | Farbe |
| options | Verpflichtend | Array(IDRef): Option | Optionen |

### Anmerkungen

Die Optionen sollten als QuickOption angelegt werden. Als phases muss OtherWorldEncountery angegeben werden

Feld muss ein OtherWorld Feld sein. Wenn kein Feld angegeben wird, gilt die Encounter für alle Welten dieser Farbe.

### JS-Übersetzung:

Create: game.createOtherWorldEncounter (obj)  
Register: game.registerOtherWorldEncounter (object)  
ID-Prefix: OE  
Constant-Scope: OtherWorldEncounter

## Headline

### Beschreibung

Eine Mythoskarte vom Typ Schlagzeile. Headlines werden direkt ausgeführt und danach verworfen.

Jede Mythoskarte bewegt Monster, erstellt Tore, und platziert Hinweismarker.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| id | Automatisch | ID | Eindeutige ID |
| type | Automatisch | Enum: Mythos | Mythos Art (Headline) |
| name | Optional | Name | Name |
| description | Verpflichtend | String | Beschreibungstext |
| shortDesc | Optional | String | Kurzbeschreibung |
| clueField | Optional | Enum: Field | Feld für Hinweismarker (zufällig wenn fehlt) |
| gateField | Optional | Enum: Field | Feld für Tor (zufällig wenn fehlt) |
| moveBlack | Optional | Array(Enum): Dimension | Dimensionen für Bewegung nach Schwarz (zufällig wenn fehlt) |
| moveWhite | Optional | Array(Enum): Dimension | Dimensionen für Bewegung nach Weiß (zufällig wenn fehlt) |
| properties | Optional | Literal | Dynamische Eigenschaften (fortgeschritten) |
| executeHeadline | Verpflichtend | Function | Funktion der Headline |

### JS-Übersetzung:

Create: game.createMythosCard(obj)  
Register: game.registerMythosCard(object)  
ID-Prefix: MY  
Constant-Scope: Mythos  
Type: Headline

## Rumor

### Beschreibung

Eine Mythoskarte vom Typ Gerücht. Es kann immer nur ein Gerücht aktiv sein. Wenn ein neues Gerücht gezogen wird, wird das neue verworfen.

Gerüchte stellen eine länger laufende Bedrohung dar. In der Regel wird in jeder Mythos Phase das Gerücht weiter fortgeschritten (z.B. Platzieren von Markern), bis ein bestimmter Wert erreicht ist. Dann haben die Spieler einen Misserfolg (Fail), und negative Auswirkungen werden aktiv.

Jedes Gerücht stellt den Spielern eine Möglichkeit, das Gerücht abzuwenden (z.B. durch Abgabe von Gegenständen, etc.), als zusätzliche Option an einem Bestimmten Feld. Wenn diese Bedingung erfüllt ist, haben die Spieler einen Erfolg (Pass), und positive Auswirkungen werden aktiv.

Jede Mythoskarte bewegt Monster, erstellt Tore, und platziert Hinweismarker.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| id | Automatisch | ID | Eindeutige ID |
| type | Automatisch | Enum: Mythos | Mythos Art (Rumor) |
| name | Optional | Name | Name |
| description | Verpflichtend | String | Beschreibungstext |
| shortDesc | Optional | String | Kurzbeschreibung |
| clueField | Optional | Enum: Field | Feld für Hinweismarker (zufällig wenn fehlt) |
| gateField | Optional | Enum: Field | Feld für Tor (zufällig wenn fehlt) |
| moveBlack | Optional | Array(Enum): Dimension | Dimensionen für Bewegung nach Schwarz (zufällig wenn fehlt) |
| moveWhite | Optional | Array(Enum): Dimension | Dimensionen für Bewegung nach Weiß (zufällig wenn fehlt) |
| properties | Optional | Literal | Dynamische Eigenschaften (fortgeschritten) |
| rumorFieldOptionId | Verpflichtend | IDRef: Option | Option für Success-Bedingung |
| rumorField | Verpflichtend | Enum: Field | Feld für Success-Option |
| setupRumor | Verpflichtend | Function | Vorbereitungsfunktion. Wird aufgerufen, wenn das Gerücht gezogen wird |
| teardownRumor | Verpflichtend | Function | Aufräumfunktion. Wird aufgerufen, wenn das Gerücht entfernt wird (Pass & Fail) |
| onMythos | Verpflichtend | Function | Wird in jeder Mythos-Phase aufgerufen, solange das Gerücht aktiv ist |
| onPass | Verpflichtend | Function | Wird aufgerufen, wenn das Gerücht abgewendet wird |
| onFail | Verpflichtend | Function | Wird aufgerufen, wenn die Spieler versagen |

### Anmerkungen

Die onMythos Funktion sollte den Status prüfen. Wenn ein Fail bzw. Pass Status erreicht wird, muss sie this.fail() bzw. this.pass() aufrufen.

Wenn der Success-Status durch die Option festgelegt wird, kann die Option mit game.context.rumor.pass(); das Gerücht abwenden.

### JS-Übersetzung:

Create: game.createMythosCard(obj)  
Register: game.registerMythosCard(object)  
ID-Prefix: MY  
Constant-Scope: Mythos  
Type: Rumor

### Beispiel

Als Beispiel eine vereinfachte Variante des SouthSideStrangler Gerüchts: Anfangs sind 7 Allies vorhanden, jede Mythosphase wird dies um 1 verringert. Wenn kein Ally mehr da ist, Fehlschlag. In Ma’s Boarding House können 5 Hinweismarker ausgegeben werden, um das Gerücht abzuwenden.

QuickOption SSSOption {  
 costs: [ Clue 5 ]; // 5 Hinweismarker für Pass  
 phases: ArkhamEncountery;  
 activate: {{{ game.context.rumor.pass(); }}};  
}  
  
Rumor SSS {  
 name: "South Side Strangler";  
 description: "…";  
 rumorFieldOptionId: Option.SSSOption;  
 rumorField: SS\_MasBoardingHouse;  
 setupRumor: {{{ this.allyCount = 7; }}}; // 7 Runden  
 teardownRumor: {{{ }}}; // Kein Teardown  
  
 onMythos: {{{   
 this.allyCount--;  
 // Fail wenn keine Allies mehr da  
 if (this.allyCount == 0) this.fail();  
 }}};  
  
 onPass: {{{  
 for (var c in game.context.allCharacters) {  
 c.addMoney(5); // Jeder Spieler erhält $5  
 }  
 }}};  
  
 onFail: {{{  
 for (var c in game.context.allCharacters) {  
 // Jeder erhält 1 Santiy & Stamia Schaden  
 c.damageSanity(1);  
 c.damageStamina(1);  
 }  
 }}};  
}

## Environment

### Beschreibung

Eine Mythoskarte vom Typ Umgebung.

Umgebungen haben Einfluss auf die Spielsituation durch Modifikatoren, oder eine Aktivität auf einem bestimmten Feld anbieten. Es kann immer nur 1 Umgebung aktiv sein. Wenn eine neue Umgebung gezogen wird, ersetzt diese die alte.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| id | Automatisch | ID | Eindeutige ID |
| type | Automatisch | Enum: Mythos | Mythos Art (Environment) |
| name | Optional | Name | Name |
| description | Verpflichtend | String | Beschreibungstext |
| shortDesc | Optional | String | Kurzbeschreibung |
| clueField | Optional | Enum: Field | Feld für Hinweismarker (zufällig wenn fehlt) |
| gateField | Optional | Enum: Field | Feld für Tor (zufällig wenn fehlt) |
| moveBlack | Optional | Array(Enum): Dimension | Dimensionen für Bewegung nach Schwarz (zufällig wenn fehlt) |
| moveWhite | Optional | Array(Enum): Dimension | Dimensionen für Bewegung nach Weiß (zufällig wenn fehlt) |
| properties | Optional | Literal | Dynamische Eigenschaften (fortgeschritten) |
| environmentType | Verpflichtend | Enum: EnvironmentType | Umgebungs-Typ |
| environmentFieldOptionId | Optional | IDRef: Option | Aktivitats-Option für Environment |
| environmentField | Verpflichtend | Enum: Field | Aktivitäts-Feld |
| environmentModifications | Optional | Complex: Mods | Modifikatoren |
| monsterModifications | Optional | Complex: MonsterMods | Monster-Spezifische Modifikatoren |
| monsterMoveModifications | Optional | Complex: Monster Move Mods | Monster Bewegungsmodifikatoren |

### JS-Übersetzung:

Create: game.createMythosCard(obj)  
Register: game.registerMythosCard(object)  
ID-Prefix: MY  
Constant-Scope: Mythos  
Type: Environment

## Investigator

### Beschreibung

Investigatoren sind die Charaktere, die die Spieler wählen können.

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| id | Automatisch | ID | Eindeutige ID |
| name | Optional | Name | Name |
| occupation | Optional | String | Beruf |
| homeFieldId | Verpflichtend | Enum: Field | Startfeld |
| sanity | Verpflichtend | Number | Geistige Gesundheit (Max) |
| stamina | Verpflichtend | Number | Ausdauer (Max) |
| money | Verpflichtend | Number | Geld (Start) |
| focus | Verpflichtend | Number | Fokus |
| clues | Verpflichtend | Number | Hinweismarker |
| fixedPossessionObjectIds | Verpflichtend | Array(IDRef): Object | Fixer Besitz |
| randomPossessions | Verpflichtend | Complex: Random Possesions | Zufälliger Besitz |
| attributes | Verpflichtend | Complex: Character Attributes | Attribute |
| uniqueAbilityId | Optional | IDRef: Object | Besondere Fähigkeit |

### Anmerkungen

Die besondere Fähigkeit muss eine UniqueAbility sein.

### JS-Übersetzung:

Create: game.createInvestigator(obj)  
Register: game.registerInvestigator (object)  
ID-Prefix: IN  
Constant-Scope: Investigator

## AncientOne

### Beschreibung

Ancient Ones sind die Großen Alten, die die Hauptwidersacher der Spieler darstellen. In jedem Spiel ist 1 Ancient One im Spiel. Während des Spiels ist der Ancient One im Schlaf. Durch verschiedene Umstände kann der AncientOne erwachen, wodurch es zum Endkampf kommt.

Ancient Ones können während dem Schlaf globale Modifikatoren haben, haben Worshippers, die oft durch Monster Modifikatoren beeinflusst werden. Im Endkampf haben sie eine Attacke, deren Schwierigkeit in der Regel mit jeder Kampfrunde steigt.

Die Investigatoren müssen Schaden akkumulieren, um den AncientOne zu besiegen. AncientOnes können hier wie normale Monster Attribute haben (z.B. Physische Resistenz).

### Attribute

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Art | Typ | Beschreibung |
| id | Automatisch | ID | Eindeutige ID |
| name | Optional | Name | Name |
| doomTrack | Verpflichtend | Number | Anzahl Doom-Marker |
| worshippersText | Optional | String | Beschreibungstext für Worshippers (links auf Karte) |
| monsterModifications | Optional | Complex: Monster Mods | Monstermodifikatoren für Worshippers |
| monsterMoveModifications | Optional | Complex: Monster Move Mods | Bewegungsmodifikatoren für Worshippers |
| powerTitle | Optional | String | Überschrift für spezielle Fähigkeit (mittig auf Karte) |
| powerText | Optional | String | Beschreibung für spezielle Fähigkeit(mittig auf Karte) |
| slumberModifications | Optional | Complex: Mods | Spezielle Fähigkeiten während des Schlafs (Power) |
| battleStartText | Optional | String | Beschreibung der Aktion wenn der Endkampf beginnt (mittig auf Karte) |
| onAwake | Optional | Function | Funktion wenn der Endkampf beginnt (AO aufwacht) |
| combatAdjustment | Optional | Number | Anpassung für Kampfprobe |
| defenses | Optional | Array(Enum): Monster Attributes | Verteidigungs-Attribute |
| attackText | Optional | String | Beschreibung der Attacke (rechts auf Karte) |
| attackAdjustment | Optional | Number | Anpassung für Attacke. Wird in der Regel jede Kampfrunde um 1 verringert |
| attack | Verpflichtend | Function | Angriff |

### JS-Übersetzung:

Create: game.createIAncientOne(obj)  
Register: game.registerAncientOne (object)  
ID-Prefix: AO  
Constant-Scope: AncientOne

### Beispiel

Hier als Beispiel der AncientOne Hastur:

AncientOne Hastur {  
 doomTrack: 13;  
// Worshippers  
/////////////////////////////////////////////////  
 worshippersText: "Hastur's worshippers ride byakhee mounts "+  
 "that they call with enchanted whistles. "+  
 "<b>Cultists</b> are <em>flying monsters</em> and "+  
 "their combat rating is -2.";  
 // Cultists are Flying  
 monsterMoveModifications: [  
 Cultist Flying  
 ];  
 // Cultists have Combat Adjustment 1. Add -3 => -2  
 monsterModifications: [  
 Cultist { Monster\_CombatAdjustment -3 }  
 ];  
// Slumber / Power  
/////////////////////////////////////////////////  
 powerTitle: "The King in Yellow";  
 powerText: "While Hastur stirs in his slumber, the cost to "+  
 "seal a gate is 8 Clue tokens instead of 5.";  
 // It costs 3 more to seal gates  
 slumberModifications: [  
 Game\_SealClueCost +3  
 ];  
// Battle Start  
///////////////////////////////////////////////////  
 battleStartText:  
 "Combat adjustment is set to the current terror level";  
 // Initial Combat Adjustment: 0  
 combatAdjustment: 0;  
 // Set combat adjustment to Terrorlevel onAwake  
 onAwake: {{{  
 this.setCombatAdjustment(game.context.terrorLevel);  
 }}};  
// Attack / End fight  
///////////////////////////////////////////////////  
attackText: "Each investigator must pass a Luck (+1) check "+  
 "or lose 2 Sanity. "+  
 "This check's modifier decreases by 1 each turn " +  
 "(+0 the 2nd turn, -1 the 3rd turn, etc.)";  
  
 defenses: PhysicalResistance;  
 // Initial Attack Adjustment is 1.   
 // Autmotically decreased each fight round  
 attackAdjustment: 1;  
 attack: {{{  
 // Roll dice:  
 var res = game.context.skillTest(  
 "Hastur attacks",   
 Constants.Skills.Luck, // Test LUCK  
 this.attackAdjustment, // Adjustment  
 1); // Required Successes  
 // If not succeeded, damage sanity by 2  
 // In Endfight, insane characters are automatically  
 // devoured, and cannot come back  
 if (!res) {  
 CHAR.damageSanity(2);  
 }  
 }}}  
}

# Enums