

# RPG-X2 Lua Dokumentation

Ubergames Walter Julius 'GSIO01' Hennecke

30. Dezember 2010



# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>7</b>
1.1	Grundlegende Informationen . . . . .	7
1.2	Vorvereinbarungen . . . . .	7
<b>2</b>	<b>Lua Hooks</b>	<b>9</b>
2.1	Was ist ein Lua Hook . . . . .	9
2.2	Statische Lua Hooks . . . . .	9
2.2.1	InitGame . . . . .	9
2.2.2	ShutdownGame . . . . .	9
2.2.3	RunFrame . . . . .	10
2.2.4	GClientPrint . . . . .	10
2.2.5	GPrint . . . . .	10
2.3	Dynamische Lua Hooks . . . . .	11
2.3.1	luaThink . . . . .	11
2.3.2	luaTouch . . . . .	11
2.3.3	luaUse . . . . .	11
2.3.4	luaHurt . . . . .	11
2.3.5	luaDie . . . . .	11
2.3.6	luaFree . . . . .	12
2.3.7	luaReached . . . . .	12
2.3.8	luaReachedAngular . . . . .	12
2.3.9	luaTrigger . . . . .	12
2.3.10	luaSpawn . . . . .	12
<b>3</b>	<b>RPG-X2 Map Scripting</b>	<b>13</b>
3.1	Map scripts . . . . .	13
3.2	Aufruf von Funktionen . . . . .	13
<b>4</b>	<b>RPG-X2 Lua Bibliotheken</b>	<b>15</b>
4.1	game . . . . .	15
4.1.1	game.Print . . . . .	15
4.1.2	game.ClientPrint . . . . .	15
4.1.3	game.CenterPrint . . . . .	15

4.1.4	game.MessagePrint . . . . .	15
4.1.5	game.LevelTime . . . . .	16
4.1.6	game.SetGlobal . . . . .	16
4.1.7	game.GetGlobal . . . . .	16
4.2	qmath . . . . .	17
4.2.1	qmath.abs . . . . .	17
4.2.2	qmath.sin . . . . .	17
4.2.3	qmath.cos . . . . .	17
4.2.4	qmath.tan . . . . .	17
4.2.5	qmath.asin . . . . .	17
4.2.6	qmath.acos . . . . .	17
4.2.7	qmath.atan . . . . .	17
4.2.8	qmath.floor . . . . .	17
4.2.9	qmath.ceil . . . . .	18
4.2.10	qmath.fmod . . . . .	18
4.2.11	qmath.modf . . . . .	18
4.2.12	qmath.sqrt . . . . .	18
4.2.13	qmath.log . . . . .	18
4.2.14	qmath.log10 . . . . .	18
4.2.15	qmath.deg . . . . .	18
4.2.16	qmath.rad . . . . .	18
4.2.17	qmath.frexp . . . . .	18
4.2.18	qmath.ldexp . . . . .	19
4.2.19	qmath.min . . . . .	19
4.2.20	qmath.max . . . . .	19
4.2.21	qmath.random . . . . .	19
4.2.22	qmath.crandom . . . . .	19
4.3	vector . . . . .	20
4.3.1	vector.New . . . . .	20
4.3.2	vector.Construct . . . . .	20
4.3.3	vector.Set . . . . .	20
4.3.4	vector.clear . . . . .	20
4.3.5	vector.Add . . . . .	20
4.3.6	vector.Substract . . . . .	20
4.3.7	vector.Scale . . . . .	20
4.3.8	vector.Length . . . . .	21
4.3.9	vector.Normalize . . . . .	21
4.3.10	vector.RotateAroundPoint . . . . .	21
4.3.11	vector.Perpendicular . . . . .	21
4.4	entity . . . . .	22
4.4.1	entity.Find . . . . .	22
4.4.2	entity.FindNumber . . . . .	22
4.4.3	entity.FindBModel . . . . .	22
4.4.4	ent.GetNumber . . . . .	22

4.4.5	ent.SetKeyValue . . . . .	22
4.4.6	entity.Remove . . . . .	22
4.4.7	ent.GetOrigin . . . . .	23
4.4.8	ent.IsClient . . . . .	23
4.4.9	ent.GetClientname . . . . .	23
4.4.10	ent.GetClassname . . . . .	23
4.4.11	ent.SetClassname . . . . .	23
4.4.12	ent.GetTargetname . . . . .	23
4.4.13	ent.SetupTrigger . . . . .	23
4.4.14	entity.GetTarget . . . . .	23
4.4.15	entity.Use . . . . .	23
4.4.16	entity.Spawn . . . . .	24
4.4.17	entiy.CallSpawn . . . . .	24
4.4.18	entity.DelayedCallSpawn . . . . .	24
4.4.19	entity.RemoveSpawns . . . . .	24
4.4.20	ent.Lock . . . . .	24
4.4.21	ent.Unlock . . . . .	24
4.4.22	ent.IsLocked . . . . .	24
4.4.23	ent.GetParm . . . . .	24
4.4.24	ent.SetParm . . . . .	25
4.5	mover . . . . .	26
4.5.1	mover.Halt . . . . .	26
4.5.2	mover.HaltAngles . . . . .	26
4.5.3	mover.AsTrain . . . . .	26
4.5.4	mover.SetAngles . . . . .	26
4.5.5	mover.SetPosition . . . . .	26
4.5.6	mover.ToAngles . . . . .	26
4.5.7	mover.ToPosition . . . . .	27
4.6	sound . . . . .	28
4.6.1	Sound Kanäle . . . . .	28
4.6.2	sound.PlaySound . . . . .	28
<b>5</b>	<b>Beispiele</b>	<b>29</b>
5.1	Beispiel 1 - Hallo Welt . . . . .	29
5.1.1	Hallo Welt für game . . . . .	29
5.1.2	Hallo Welt für einen Spieler . . . . .	29
5.1.3	Hallo Welt für alle Spieler . . . . .	30
5.2	Beispiel 2 - Entities Finden . . . . .	30
5.2.1	Entities über ihren targetname finden . . . . .	31
5.2.2	Entities über ihre Entitynummer finden . . . . .	31
5.2.3	Entities über ihr Brush Modell finden . . . . .	31
5.3	Beispiel 3 - Entities Spawnen . . . . .	31

<b>6</b>	<b>Wie man ...</b>	<b>35</b>
6.1	Turbolifte zu älteren RPG-X Maps hinzufügt . . . . .	35
6.2	Transporter die das ui_transporter benutzen zu älteren Maps hinzufügt . . . . .	35
6.3	func_usable zu func_forcefield konvertiert . . . . .	35

# Kapitel 1

## Einführung

### 1.1 Grundlegende Informationen

Die RPG-X2 Lua Dokumentation dokumentiert und beschreibt alle Lua Funktionen die in RPG-X2 verfügbar sind. Die version die Sie gerade lesen ist für die **RPG-X2 Version 2.2 beta 4.4.5**. Die RPG-X2 Lua Dokumentation wird mit dem Erscheinen jeder neuen RPG-X Version aktualisiert.

### 1.2 Vorvereinbarungen

- In Lua werden Variablen nicht mit ihrem Typ deklariert. Um Sie dennoch über den Typ einer Variablen informieren zu können werden die Typangaben Kursiv vor die Variablen geschrieben (Beispiel: *integer clientNum*).
- Es gibt in RPG-X2 Lua drei verschiedene Funktionsaufrufe.
  - Funktionsaufrufe der Lua Basis Bibliotheken (Beispiel: **tostring(clientNum)**).
  - Funktionsaufrufe der RPG-X2 Bibliotheken, welche den Bibliotheksnamen vorangestellt haben (Beispiel: **entity.Spawn()**).
  - Funktionsaufrufe über Variablen. Dies geht zum Beispiel mit Entities und Vektoren (Beispiel: **ent.Remove(ent)**).
  - Funktionsaufrufe über Variablen bei denen die Variable selber das erste Argument ist **var.function(var)** können auch als **var:function()** geschrieben werden (Beispiel: **ent.Remove(ent)** is the same as **ent:Remove()**).



# Kapitel 2

## Lua Hooks

### 2.1 Was ist ein Lua Hook

Ein Lua Hook ist eine Funktion die aufgerufen wird wenn ein spezifisches Ereignis in der Spiellogik eintritt. So wird zum Beispiel beim initialisieren der Spiellogik die Funktion `G_InitGame` aufgerufen. Diese Funktion besitzt einen Lua Hook, was bedeutet das beim Aufruf von `G_InitGame` auch eine Lua Funktion aufgerufen wird. In RPG-X2 Lua gibt es Lua Hooks mit festen und Lua Hooks mit dynamischen Namen.

### 2.2 Statische Lua Hooks

Statische Lua Hooks haben einen festen Namen.

#### 2.2.1 InitGame

**InitGame**(*integer leveltime*, *integer randomssed*, *integer restart*)

Wird nach dem Spielstart oder nach dem benutzen des `map_restart` Kommandos aufgerufen.

**leveltime** die aktuelle Levelzeit in Millisekunden

**restart** ist 1 bei einem `map_restart` und ansonsten 0

#### 2.2.2 ShutdownGame

**ShutdownGame**(*integer restart*)

Wird beim Beenden des Spiels aufgerufen, dies kann sein bei `disconnect`, schließen des Spiels, Map wechsel, `map_restart`.

**restart** ist 1 bei einem `map_restart` und ansonsten 0

### 2.2.3 RunFrame

**RunFrame**(*integer* **leveltime**)

Wird jeden Frame aufgerufen. Diese Funktion sollte wenn dann nur mit äußerster Vorsicht verwendet werden, da Zeit zwischen Frames nur 50 ms beträgt.

**leveltime** aktuelle Levelzeit in Millisekunden

### 2.2.4 GClientPrint

**GClientPrint**(*string* **text**, *entity* **client**)

Wird beim Aufruf der Spiellogikfunktion G.PrintfClient aufgerufen.

**text** der ausgegeben wird

**client** Client bei dem der Text ausgegeben wird.

### 2.2.5 GPrint

**GPrint**(*string* **text**)

Wird beim Aufruf der Spiellogikfunktion G.Print aufgerufen.

**text** der in die Spielkonsole ausgegeben wird. (Achtung die Spielkonsole ist die des Servers nicht die des Client!)

## 2.3 Dynamische Lua Hooks

Diese Lua Hook können verschiedene Namen haben. Sie sind alle Lua Hooks für Entities. Die Funktionsnamen für diese Lua Hooks werden im Radiant mit Hilfe von Schlüssel-Wert-Paaren definiert. Da die Funktionsnamen dieser Lua Hooks von diesen Paaren abhängen werden in dieser Dokumentation die Schlüssel als Funktionsnamen verwendet, die den Namen der entsprechenden Funktion enthalten.

### 2.3.1 luaThink

**luaThink**(*entity ent*)

Wird jedes mal aufgerufen wenn die Entity denkt.  
**ent** die Entity selbst

### 2.3.2 luaTouch

Wird jedes mal aufgerufen wenn die Entity berührt wird.  
**ent** die Entity selbst  
**other** die Entity welche die Entity **ent** berührt hat.

### 2.3.3 luaUse

**luaUse**(*entity ent*), *entity other*, *entity activator*)

Wird jedes mal aufgerufen wenn die Entity benutzt wird.  
**ent** die entity selbst  
**activator** die Entity welche die Entity **ent** benutzt hat.

### 2.3.4 luaHurt

**luaHurt**(*entity ent*, *entity inflictor*, *entity attacker*)

Wird jedes mal aufgerufen wenn der Entity Schaden zugefügt wird.  
**ent** die Entity selbst  
**inflictor** der Infliktor  
**attacker** der Angreifer

### 2.3.5 luaDie

**luaDie**(*entity ent*, *entity inflictor*, *entity attacker*, *integer dmg*, *integer mod*)

Wird jedes mal aufgerufen wenn die Entity stirbt.  
**ent** die Entity selbst  
**inflictor** der Infliktor  
**attacker** der Angreifer  
**dmg** menge des Schadens  
**mod** Gründe des Todes

### 2.3.6 luaFree

#### **luaFree**(*entity ent*)

Wird aufgerufen wenn die Entity freigegeben wird was gleichbedeutend mit deren Löschung ist.

**ent** die Entity selbst

### 2.3.7 luaReached

#### **luaReached**(*entity ent*)

Wird aufgerufen wenn die Bewegung einer Entity ihren Endpunkt erreicht.

**ent** die Entity selbst

### 2.3.8 luaReachedAngular

#### **luaReachedAngular**(*entity ent*)

Wird aufgerufen wenn die Drehbewegung einer Entity ihren Endpunkt erreicht.

**ent** die Entity selbst

### 2.3.9 luaTrigger

#### **luaTrigger**(*entity ent, entity other*)

Wird aufgerufen wenn eine Entity getriggert wird. Man beachte das dies nicht das gleiche ist wie die Benutzung einer Entity. Dieser Lua Hook ist für `trigger_*` Entities gedacht.

**ent** die Entity selbst

**other** die Entity die das Triggerereignis ausgelöst hat

### 2.3.10 luaSpawn

#### **luaSpawn**(*entity ent*)

Wird aufgerufen wenn die Spawnfunktion einer Entity aufgerufen wird.

**ent** die Entity selbst

## Kapitel 3

# RPG-X2 Map Scripting

### 3.1 Map scripts

Im moment kann genau ein Scriptdatei je Map geladen werden. Diese Scriptdatei muss im Ordner *scripts/lua/jmapname* sein und den Namen *jmapname.lua* haben.

### 3.2 Aufruf von Funktionen

Es gibt dynamische Lua Hooks für die Verwendung im Radiant (siehe Dynamische Lua Hooks). Sie könne diese Lua Hooks für Entites verwenden, in dem Sie das entsprechende Schlüssel-Wert-Paar zu der Entity hinzufügen. Soll zum Beispiel eine Funktion *PrintText* aufgerufen werden wenn eine *func\_usable* benutzt wird so müssen die zu der Entity die Schlüssel *luaUse* und den Wrt *PrintText* hinzufügen.



# Kapitel 4

## RPG-X2 Lua Bibliotheken

### 4.1 game

Diese Bibliothek ermöglicht Zugriff auf einige Spiellogikfunktionen wie zum Beispiel `G_Printf` und `G_ClientPrintf`.

#### 4.1.1 game.Print

**game.Print**(*string text*)

Gibt **text** in der Spielkonsole(Serverkonsole) aus.

#### 4.1.2 game.ClientPrint

**game.ClientPrint**(*integer clientNum, string text*)

Gibt **text** in der Konsole des Client mit der Clientnummer **clientNum** aus. Falls **clientNum** gleich -1 ist wird der Text in den Konsolen aller Spieler ausgegeben.

#### 4.1.3 game.CenterPrint

**game.CenterPrint**(*integer clientNum, string text*)

Gibt **text** auf der Mitte des Bildschirms des Spielers mit der Clientnummer **clientNum** aus. Falls **clientNum** gleich -1 ist erfolgt die Ausgabe bei allen Spielern.

#### 4.1.4 game.MessagePrint

**game.MessagePrint**(*integer clientNum, string text*)

Gibt **text** in der rechten unteren Ecke des Bildschirms des Spielers mit der Clientnummer **clientNum** aus. Falls **clientNum** gleich -1 ist erfolgt die Ausgabe bei allen Spielern.

#### 4.1.5 `game.LevelTime`

`game.LevelTime()` Gibt die aktuelle Levelzeit in Millisekunden zurück.

#### 4.1.6 `game.SetGlobal`

`game.SetGlobal(string name, value)`

Setzt eine globale Lua Variable mit den Namen **name** und dem Wert **value**, welche dann über alle Funktionen hinweg verfügbar ist.

#### 4.1.7 `game.GetGlobal`

`game.GetGlobal(string name)`

Gibt den Wert eine globalen Variable mit dem Namen **name** zurück falls diese existiert.

## 4.2 qmath

Diese Bibliothek ermöglicht den Zugriff auf mehrere mathematische Funktionen die in der Spiellogik verfügbar sind.

### 4.2.1 qmath.abs

**qmath.abs**(*float* f)

Gibt den ganzzahligen Wert von **f** zurück.

### 4.2.2 qmath.sin

**qmath.sin**(*float* degree)

Gibt den sinus von **degree** zurück.

### 4.2.3 qmath.cos

**qmath.cos**(*float* degree)

Gibt den cosinus von **degree** zurück.

### 4.2.4 qmath.tan

**qmath.tan**(*float* degree)

Gibt den tangenz von **degree** zurück.

### 4.2.5 qmath.asin

**qmath.asin**(*float* f)

Gibt den arsinus von **f** zurück.

### 4.2.6 qmath.acos

**qmath.acos**(*float* f)

Gibt den arcosinus von **f** zurück.

### 4.2.7 qmath.atan

**qmath.atan**(*float* f)

Gibt den artangenz von **f** zurück.

### 4.2.8 qmath.floor

**qmath.floor**(*float* f)

Gibt den abgerundeten Wert von **f** zurück.

#### 4.2.9 `qmath.ceil`

`qmath.ceil(float f)`

Gibt den aufgerundeten Wert von `f` zurück.

#### 4.2.10 `qmath.fmod`

`qmath.fmod(float f, float n)`

Gibt den Rest von  $f/n$ . zurück.

#### 4.2.11 `qmath.modf`

`qmath.modf(float f)`

Zerlegt  $f$  in einen ganzzahligen und einen fraktionales Teil. Der fraktionales Teil wird zurückgegeben und der ganzzahlige Teil in  $f$  gespeichert.

#### 4.2.12 `qmath.sqrt`

`qmath.sqrt(float f)`

Gibt die Wurzel aus `f` zurück.

#### 4.2.13 `qmath.log`

`qmath.log(float f)`

Gibt den Logarithmus von `f` zurück.

#### 4.2.14 `qmath.log10`

`qmath.log10(float f)`

Gibt den Logarithmus von `f` zur Basis 10 zurück.

#### 4.2.15 `qmath.deg`

`qmath.deg(float radian)`

Konvertiert Bogenmaß zu grad.

#### 4.2.16 `qmath.rad`

`qmath.rad(float degree)`

Konvertiert Grad zu Bogenmaß.

#### 4.2.17 `qmath.frexp`

`qmath.frexp(float f)`

Zerlegt `f` in seine binäre Signifikante und einen integralen Exponenten von 2, so dass gilt:

$x = \text{Signifikante} * 2^{\text{Exponent}}$

**4.2.18 qmath.ldexp****qmath.ldexp**(*float* **f**, *float* **n**)Gibt das Resultat der Multiplikation von **f** mit 2 potenziert mit **n** zurück.**4.2.19 qmath.min****qmath.min**(*integer* **array**[])Gibt **array**[].**4.2.20 qmath.max****qmath.max**(*integer* **array**[])Gibt den höchsten Wert aus **array**[] zurück.**4.2.21 qmath.random****qmath.random**()

Gibt zufällige ganzzahlige Werte zurück.

**4.2.22 qmath.crandom****qmath.crandom**()

Gibt zufällige Gleitkommazahlen zurück (Bei der Generierung wird die sogenannte crazy random function verwendet).

### 4.3 vector

Diese Bibliothek implementiert einen neuen Variablentyp `vector` sowie mathematische Funktionen zum Rechnen mit Vektoren.

#### 4.3.1 vector.New

`vector.New()`

Erzeugt einen neuen Vektor  $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ .

#### 4.3.2 vector.Construct

`vector.Construct(float x, float y, float z)`

Erzeugt einen neuen Vektor  $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ .

#### 4.3.3 vector.Set

`vector.Set(vector v, float x, float y, float z)`

Setzt einen Vektor `v` auf die angegebenen Werte.

#### 4.3.4 vector.clear

`vector.Clear(vector v)`

Säubert einen `vector` indem er auf  $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$  gesetzt wird.

#### 4.3.5 vector.Add

`vector.Add(vector a, vector b, vector c)`

Bildet die Summe von `a` und `b` und speichert das Ergebnis in `c`.

#### 4.3.6 vector.Subtract

`vector.Subtract(vector a, vector b, vector c)`

Subtrahiert `b` von `a` und speichert das Ergebnis in `c`.

#### 4.3.7 vector.Scale

`vector.Scale(vector a, float b, vector c)`

Skaliert `a` um den Faktor `b` und speichert das Ergebnis in `c`.

#### 4.3.8 `vector.Length`

`vector.Length(vector a)`

Gibt die Länge von **a** zurück.

#### 4.3.9 `vector.Normalize`

`vector.Normalize(vector a)`

Normalisiert **a**.

#### 4.3.10 `vector.RotateAroundPoint`

`vector.RotateAroundPoint(vector dest, vector dir, vector point, float degrees)`

Rotiert einen Punkt **point** um einen gegebene Vektor.

**dir** Normalisierter Vektor um den rotiert werden soll.

**point** Punkt der rotiert werden soll

**degrees** um wieviel Grad rotiert werden soll

**dst** der Punkt **point** nach der Drehung

#### 4.3.11 `vector.Perpendicular`

`vector.Perpendicular(vector dest, vector src)`

Findet einen zum Quellvektor senkrechten Zielvektor. **src** Quellvektor **dest**

Ein zum Quellvektor senkrechter Vektor (das Ergebnis)

## 4.4 entity

Diese Bibliothek enthält Funktionen für Entities. Alle Funktionen mit vorangestellten *entity* sind Funktionsaufrufe über die Bibliothek. Alle Funktionen mit vorangestellten *ent* sind Funktionsaufrufe auf einer Entity.

### 4.4.1 entity.Find

**entity.Find**(*string* **targetname**)

Gibt die erste gefundene Entity zurück deren **targetname** dem gesuchten **targetname** entspricht.

### 4.4.2 entity.FindNumber

**entity.FindNumber**(*integer* **entnum**)

Gibt die Entity mit der Entitynummer **entnum** zurück.

### 4.4.3 entity.FindBModel

**entity.FindBModel**(*integer* **bmodelnum**)

Gibt die Entity mit dem brush model **\*bmodelnumber** zurück. Dies ist der einzig sichere weg eine Brushentity ohne **targetname** zu finden, da sich die Entitynummern abhängig davon verändern ob eine Map lokal geladen wird, auf einen lokalen server läuft oder auf einem dedizierten Internetserver läuft ändern.

### 4.4.4 ent.GetNumber

**ent.GetNumber**(*entity* **ent**) or **ent:GetNumber**()

Gibt die Entitynummer einer Entity zurück.

### 4.4.5 ent.SetKeyValue

**ent.SetKeyValue**(*entity* **ent**, *string* **key**, *string* **value**) or **ent:SetKeyValue**(*string* **key**, *string* **value**)

Setzt ein Schlüssel-Wert-Paar für **ent** wie im Radiant. Dies funktioniert aber nur wenn der Schlüssel *key* ein Teil von *fields\_t* ist, welche die vordefinierten Schlüssel enthält.

### 4.4.6 entity.Remove

**entity.Remove**(*entity* **ent**)

Entfernt die Entity **ent**.

#### 4.4.7 ent.GetOrigin

**ent.GetOrigin**(*entity ent*) or **ent:GetOrigin**()

Gibt die Origin von **ent** als vector zurück.

#### 4.4.8 ent.IsClient

**ent.IsClient**(*entity ent*) or **ent:IsClient**()

Gibt einen boolean zurück. Ist wahr wenn **ent** ein Spieler ist.

#### 4.4.9 ent.GetClientname

**ent.GetClientname**(*entity ent*) or **ent:GetClientname**()

Gibt den clientname von **ent** zurück.

#### 4.4.10 ent.GetClassname

**ent.GetClassname**(*entity ent*) or **ent:GetClassname**()

Gibt den Klassennamen von **ent** zurück.

#### 4.4.11 ent.SetClassname

**ent.SetClassname**(*entity ent, string classname*) or

**ent:SetClassname**(*string classname*)

Setzt den Klassennamen von **ent** auf **classname**.

#### 4.4.12 ent.GetTargetname

**ent.GetTargetname**(*entity ent*) or **ent:GetTargetname**()

Gibt den targetname von **ent** zurück.

#### 4.4.13 ent.SetupTrigger

**ent.SetupTrigger**(*entity ent*) or **ent:SetupTrigger**()

Grundlegende Initialisierung für im Scripting gespannte trigger\_\* Entities.

#### 4.4.14 entity.GetTarget

**entity.GetTarget**(*entity ent*) Gibt das target von **ent** zurück.

#### 4.4.15 entity.Use

**entity.Use**(*entity ent*)

Benutzt **ent**.

#### 4.4.16 **entity.Spawn**

##### **entity.spawn()**

Versucht eine neue Entity zu spawnen und gibt diese bei Erfolg zurück, sonst wird *nil* zurückgegeben.

#### 4.4.17 **entity.CallSpawn**

##### **entity.CallSpawn(*entity ent*)**

Ruft die Spawnfunktion der Spiellogik für **ent** auf.

#### 4.4.18 **entity.DelayedCallSpawn**

##### **entity.DelayedCallSpawn(*entity ent*, *integer delay*)**

Tut das gleiche wie CallSpawn wartet aber eine mit **delay** festgelegte Zeit bis die Funktion aufgerufen wird. **delay** Zeit in Millisekunden die gewartet werden soll.

#### 4.4.19 **entity.RemoveSpawns**

##### **entity.RemoveSpawns()**

Löscht alle Spawnpunkte der Map.

#### 4.4.20 **ent.Lock**

##### **ent.Lock(*entity ent*)**

Sperrt/Verschließt die Entity. Funktioniert mit allen Entities die gesperrt/-verschlossen werden können (Türen, Turbolifte, Usables, ...).

#### 4.4.21 **ent.Unlock**

##### **ent.Unlock(*entity ent*)**

Entsperrt/Öffnet die Entity. Funktioniert mit allen Entities die gesperrt/-verschlossen werden können (Türen, Turbolifte, Usables, ...).

#### 4.4.22 **ent.IsLocked**

##### **ent.IsLocked(*entity ent*)**

Gibt **wahr** zurück falls eine entity Gesperrt/Verschlossen ist ansonsten wird **falsch** zurückgegeben.

#### 4.4.23 **ent.GetParm**

##### **ent.GetParm(*entity ent*, *integer parm*)**

Gibt den Wert für einen der luaParms der Entity **ent** als string zurück. Sollte der Wert nicht gesetzt sein wird **nil** zurückgegeben.  
**parm** der gewünschte Wert (1 bis 4)

#### 4.4.24 ent.SetParm

**ent.SetParm**(*entity* **ent**, *integer* **parm**, *string* **value**)

Setz einen gewünschten luaParm auf den Wert **value**.

**parm** gewünschter parm der gesetzt werden soll

**value** Wert auf den der parm gesetzt werden soll

## 4.5 mover

Wichtige Bemerkung: Sie sollten immer `mover.Halt` bzw. `mover.HaltAngles` aufrufen bevor Sie einen `mover` wieder durch Aufruf einer entsprechenden Funktion bewegen. Ansonsten wird die Bewegung der Entity nicht korrekt funktionieren.

### 4.5.1 mover.Halt

**mover.Halt**(*entity ent*)

Stoppt sofort jegliche Bewegung (ausgenommen Drehungen).

### 4.5.2 mover.HaltAngles

**mover.HaltAngles**(*entity ent*)

Stoppt sofort jegliche Drehung.

### 4.5.3 mover.AsTrain

**mover.AsTrain**(*entity mover, entity target, float speed*)

Bewegt die Entity wie ein *func.train*. Ziel muss ein *path\_corner* sein.

**target** die erste *path\_corner*

### 4.5.4 mover.SetAngles

**mover.SetAngles**(*entity ent, vector angles*) or **mover.SetAngles**(*entity ent, float x, float y, float z*)

Setzt die *angles* von **ent** zu dem angegebenen Wert(en).

### 4.5.5 mover.SetPosition

**mover.SetPosition**(*entity ent, vector pos*) or **mover.SetPosition**(*entity ent, float x, float y, float z*)

Set die Origin von **ent** zu den angegebenen Wert(en) und bewegt sie sofort dort hin.

### 4.5.6 mover.ToAngles

**mover.ToAngles**(*entity ent, float speed, vector angles*) or **mover.ToAngles**(*entity ent, float speed, float x, float y, float z*)

Rotiert **ent** zu den angegebenen *angles*.

#### 4.5.7 mover.ToPosition

**mover.ToPosition**(*entity* ent, *float* speed, *vector* angles) or

**mover.ToPosition**(*entity* ent, *float* speed, *float* x, *float* y, *float* z)

Bewegt **ent** zu angegebenen Position.

## 4.6 sound

Diese Bibliothek fügt die Möglichkeit hinzu Sounds abzuspielen sowie zu Verwalten.

### 4.6.1 Sound Kanäle

Einige Funktionen dieser Bibliothek haben einen Audiokanal als Parameter. Im Normalfall wird es reichen den Kanal `CHAN_AUTO` zu benutzen der die Engine die Auswahl treffen lässt. Dennoch besteht die Möglichkeit den Kanal selbst zu wählen.

Hier ist eine Tabelle mit den verschiedenen Kanälen und ihrer Werte für den Funktionsparameter:

<code>CHAN_AUTO</code>	0
<code>CHAN_LOCAL</code>	1
<code>CHAN_WEAPON</code>	2
<code>CHAN_VOICE</code>	3
<code>CHAN_ITEM</code>	4
<code>CHAN_BODY</code>	5
<code>CHAN_LOCAL_SOUND</code>	6
<code>CHAN_ANNOUNCER</code>	7
<code>CHAN_MENU1</code>	8

### 4.6.2 `sound.PlaySound`

**`sound.PlaySound`**(*entity* **ent**, *integer* **chan**, *string* **sound**)

Versucht eine Audiodatei **sound** auf dem Kanal **chan** auf der Entity **ent** abzuspielen.

# Kapitel 5

## Beispiele

Dieses Kapitel der Dokumentation enthält Beispiele welche dazu hilfreich sein könnten einige Funktionen besser zu verstehen.

### 5.1 Beispiel 1 - Hallo Welt

Das ist ein Beispiel das einfach bei jeder Programmiersprache Pflicht ist.

#### 5.1.1 Hallo Welt für game

Listing 5.1: Hallo Welt für game

```
1 function HelloWorld(ent, other, activator)
2     game.Print("Hello World");
3 end
```

Wie Sie vielleicht erkennen ist die eine Funktion für luaUse (Man kann das am Funktionskopf sehen).

#### 5.1.2 Hallo Welt für einen Spieler

Listing 5.2: Hallo Welt für Spieler

```
1 function HelloWorld(ent, other, activator)
2     game.ClientPrint(activator:GetNumber(),
3         "Hello " .. activator:GetClientname());
4     game.CenterPrint(activator:GetNumber(),
5         "Hello " .. activator:GetClientname());
6     game.MessagePrint(activator:GetNumber(),
7         "Hello " .. activator:GetClientname());
8 end
```

Wie Sie vielleicht erkennen ist die eine Funktion für luaUse (Man kann das am Funktionskopf sehen).

Listing 5.3: First function

```

1 game.ClientPrint(activator:GetNumber(),
2     "Hello_" .. activator:GetClientname());

```

Diese Funktion gibt eine Nachricht in die Spielerkonsole aus.  
 activator:GetNumber() holt die Entitynummer zurück welches in diesem Fall auch die Clientnummer ist.  
 activator:GetClientname() holt den Namen des Spielers

Listing 5.4: Second function

```

1 game.CenterPrint(activator:GetNumber(),
2     "Hello_" .. activator:GetClientname());

```

Diese Funktion gibt eine Nachricht auf der Mitte des Bildschirm eines Spielers aus.

Listing 5.5: Third function

```

1 game.MessagePrint(activator:GetNumber(),
2     "Hello_" .. activator:GetClientname());

```

Diese Funktion gibt eine Nachricht in der unteren rechten Ecke des Bildschirms eines Spielers aus.

### 5.1.3 Hallo Welt für alle Spieler

Listing 5.6: Hallo Welt für alle Spieler

```

1 function HelloWorld(ent, other, activator)
2     game.ClientPrint(-1, "Hello_all");
3     game.CenterPrint(-1, "Hello_all");
4     game.MessagePrint(-1, "Hello_all");
5 end

```

Dieses Beispiel ähnelt dem vorherigen sehr stark, der einzige Unterschied liegt darin das anstatt einer Clientnummer die -1 das erste argument ist. Dies führt dazu das die Nachricht für alle Spieler ausgegeben wird.

## 5.2 Beispiel 2 - Entities Finden

Diese Beispiele werden die verschiedenen Wege zeigen eine Entity zu finden.

### 5.2.1 Entities über ihren targetname finden

Listing 5.7: Entity über ihren targetname finden

```
1 function Example ()  
2     local ent ;  
3     ent = entity .Find (" doorbell " );  
4 end
```

Zu beachten ist das `entity.Find()` immer nur die erste Entity zurück gibt die gefunden wird. Das bedeutet das wenn es mehrere Entities mit dem selben targetname gibt es passieren kann das Sie die Entity die Sie suchen auf diese weiße nicht finden können.

Neben dem Suchen nach einer Entity zeigt dieses Beispiel auch wie man lokale Variablen benutzt.

### 5.2.2 Entities über ihre Entitynummer finden

Listing 5.8: Entities über ihre Entitynummer finden

```
1 function Example ()  
2     local ent ;  
3     ent = entity .FindNumber (22) ;  
4 end
```

Dies ist ein fast absolut sicherer Weg eine Entity zu finden, man muss aber folgendes beachten: Die Entitynummer für eine Entity ist unterschiedlich je nachdem ob man eine map lokal oder auf einem Server lädt.

### 5.2.3 Entities über ihr Brush Modell finden

Listing 5.9: Entities über ihr Brush Modell finden

```
1 function Example ()  
2     local ent ;  
3     ent = entity .FindBModel (22) ;  
4 end
```

Absolut eindeutiger Weg eine Entity zu finden, funktioniert allerdings nur für Brushentities.

## 5.3 Beispiel 3 - Entities Spawnen

Dieses Beispiel zeigt wie man Entities über das Scripting spawnen kann. Es ist möglich fast alle Entities zu spawnen die kein Brush Modell haben. Darüber hinaus kann man einige Entities spawnen die Brush Modells benötigen sofern diese nicht sichtbar sind (zum Beispiel `trigger_*` Entities).

Listing 5.10: Eine Entity Spawnen

```
1 function Example()  
2     local ent = entity.Spawn()  
3     if ent == nil then return;  
4     ent:SetKeyValue("classname", "info_notnull");  
5     mover:SetPosition(0, 0, 0);  
6     entity.CallSpawn(ent);  
7 end
```

Also was wird getan und warum?

```
1 local ent = entity.Spawn()
```

Versucht eine neue Entity zu spawnen und sie ent zuzuweisen.

```
1 if ent == nil then return
```

Dies überprüft ob das Spawnen der Entity erfolgreich war. Sollte dies nicht der Fall sein wird die weitere Abarbeitung der Funktion verhindert.

```
1 ent:SetKeyValue("classname", "info_notnull");
```

Setzt den Klassennamen der Entity und macht sie damit zu einer Entity eines bestimmten Typs.

```
1 mover:SetPosition(0, 0, 0);
```

Setzt die Origin der Entity.

```
1 entity.CallSpawn(ent);
```

Sorgt für den Aufruf der Spawnfunktion der Entity in der Spiellogik.



# Kapitel 6

## Wie man ...

### 6.1 Turbolifte zu älteren RPG-X Maps hinzufügt

Kommt demnächst ...

### 6.2 Transporter die das `ui_transporter` benutzen zu älteren Maps hinzufügt

Kommt demnächst ...

### 6.3 `func_usable` zu `func_forcefield` konvertiert

Hier wird gezeigt wie man eine `func_usable` in ein `func_forcefield` konvertiert. Bevor wir beginnen müssen wir aber erstmal einige Dinge herausfinden:

- Wie man eine `func_usable` garantiert und fehlerfrei findet.
- Wie die momentanen Spawnfalgs der Entity sind.

An diese informationen kommt man wie folgt:

- RPG-X2 starten und die Map laden.
- Als Admin einloggen oder in die Adminklasse wechseln.
- Zur `func_usable` gehen und sicherstellen das die sichtbar ist.
- Sie mit den Fadenkreuz anvisieren.
- Konsole öffnen und folgendes Kommando eingeben: **`getEntInfo`**.

Man bekommt eine Liste mit nützlichen Informationen. Sollte die Entity einen `targetname` muss man überprüfen ob sie die einzige mit diesem Namen

ist. Das tut man indem man **getEntByTargetname** gefolgt vom targetname eingibt. Falls nur eine Entity aufgelistet wird heißt dies das es nur diese eine Entity mit diesem targetname gibt. Damit sind wir fertig mit diesem Schritt.

Sollte es aber mehrere Entities mit diesem targetname geben verwendet man das Brush Modell der Entity um sie zu finden. Der Name des Brush Modells wurde bereits beim ausführen von **getEntInfo** angezeigt.

Der nächste Schritt ist es die Spawnflags dahingehen zu überprüfen ob sie dem entsprechen was man für das func\_forcefield braucht. Was zu tun ist für den Fall das die Spawnflags nicht passen werden wir unten sehen.

Nun fangen wir mit dem Scripting an. Der beste platz um eine Entitykonversion durchzuführen ist **InitGame**, da diese Funktion bereits bei jedem Mapanfang ausgeführt wird.

Listing 6.1: Beispiel 1

```

1 function InitGame(levelTime, randomSeed, restart)
2     — adjust the targetname
3     local ent = entity.Find("forcefield1");
4     if ent == nil then return;
5     ent:SetKeyValue("classname", "func_forcefield");
6     — setting the spawnflags is optional
7     — only change them if you have to
8     ent:SetKeyValue("spawnflags", "0")
9     entity.CallSpawn(ent);
10 end

```

Listing 6.2: Beispiel 2

```

1 function InitGame(levelTime, randomSeed, restart)
2     — adjust the model number
3     local ent = entity.FindBModel(22);
4     if ent == nil then return;
5     ent:SetKeyValue("classname", "func_forcefield");
6     — setting the spawnflags is optional
7     — only change them if you have to
8     ent:SetKeyValue("spawnflags", "0")
9     entity.CallSpawn(ent);
10 end

```